The Dissertation Committee for Ashish Kumar Singh Certifies that this is the approved version of the following dissertation:

Statistical Algorithms for Circuit Synthesis under Process Variation and High Defect Density

Committee:

Michael Orshansky, Supervisor

Adnan Aziz

Sani Nassif

Sanjay Shakkottai

Nur Touba

Statistical Algorithms for Circuit Synthesis under Process Variation and High Defect Density

by

Ashish Kumar Singh, B.Tech; M.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

December 2007

Dedication

To my spiritual master Srila Bhakti Anand Trivikram Gosvami Maharaj

Acknowledgements

I would first like to thank my advisor, Prof. Michael Orshansky for providing constant guidance, critical feedback and inspiration for this work. I have been very fortunate to work with Dr. Sani Nassif on two papers in timing analysis. I have learned a lot from him and I truly appreciate his support. During my Ph.D. years, I had frequent discussions with Anand Ramalingam and Murari Mani who had been extremely helpful and encouraging.

And lastly, I acknowledge the support of my parents who have supported me steadfastly in all my endeavors.

Statistical Algorithms for Circuit Synthesis under Process Variation and High Defect Density

Publication No._____

Ashish Kumar Singh, Ph.D.

The University of Texas at Austin, 2007

Supervisor: Michael Orshansky

As the technology scales, there is a need to develop design and optimization algorithms under various scenarios of uncertainties. These uncertainties are introduced by process variation and impact both delay and leakage. For future technologies at the end of CMOS scaling, not only process variation but the device defect density is projected to be very high. Thus realizing error tolerant implementation of Boolean functions with minimal redundancy overhead remains a challenging task. The dissertation is concerned with the challenges of low-power and area digital circuit design under high parametric variability and high defect density.

The technology mapping provides an ideal starting point for leakage reduction because of higher structural freedom in the choices of implementations. We first describe an algorithm for technology mapping for yield enhancement that explicitly takes parameter variability into account. We then show how leakage can be reduced by accounting for its dependence on the signal state, and develop a fast gain-based

technology mapping algorithm. In some scenarios the state probabilities can not be precise point values but are modeled as an interval. We extended the notion of mean leakage to the worst case mean leakage which is defined as the sum of maximal mean leakage of circuit gates over the feasible probability realizations. The gain-based algorithm has been generalized to optimize this proxy leakage metric by casting the problem within the framework of robust dynamic programming. The testing is performed by selecting various instance probabilities for the primary inputs that are deviations from the point probabilities with respect to which a point probability based gain based mapper has been run. We obtain leakage improvement for certain test probabilities with the interval probability based over the point probability based mapper.

Next, we present techniques based on coding theory for implementing Boolean functions in highly defective fabrics that allow us to tolerate errors to a certain degree. The novelty of this work is that the structure of Boolean functions is exploited to minimize the redundancy overhead.

Finally we have proposed an efficient analysis approach for statistical timing, which can correctly propagate the slope in the path-based statistical timing analysis. The proposed algorithm can be scaled up to one million paths.

Table of Contents

List of Tables

List of Figures

Chapter 1. Introduction

This dissertation aims to contribute to the solution of several challenges in the computer-aided design of ICs in the nanometer scale of CMOS technology and addresses the challenges of low-power digital circuit design under high parametric variability and high defectivity. Both of these fundamental challenges and the solutions we propose are overviewed in this chapter.

One of the defining features of CMOS technology as it approaches the fundamental limit of scaling is the variability in the device parameters. With scaling, the variation of device parameters does not scale at the same rate due to the manufacturing limitations. Another reason is that fundamental atomic scale randomness is becoming significant at very small device dimensions. Moreover, the composition of variability changes: the ratio of the with-in die variation, which affects even the devices on the same die diversely, with respect to the die-to-die variation has been growing [1]. For 0.13um CMOS technologies, the percentage of the with-in die component of gate length for the effective MOS channel length is 35%. The rise in with-in die variation is due to the fundamental limitations that arise when trying to operate the devices at a scale which is comparable to the atomic structure of materials. One example is the density of dopants which is different in different channels because of random placement of dopant atoms. Another example is the local oxide thickness variation that contributes to intrinsic threshold voltage fluctuation. In technologies below 32nm, Vth variation due to oxide thickness may become comparable to that due to dopant density variations. Sub-wavelength photolithography makes Lgate variation difficult to contain because of optical proximity effects. Thus for transistors having smaller size compared to the

wavelength of light, the light is affected by diffraction from neighboring transistors. Therefore the dimension of transistors is influenced by the local layout.

These trends result in the situation when the traditional means of dealing with variability (corner-based techniques) for both optimization and analysis become overly pessimistic. A related limitation of deterministic techniques is that the set of nearly critical paths can not be identified with the corner based analysis [101]. Addressing this problem with the traditional deterministic algorithms, would lead to an explosion of computational complexity because of the high number of process corners that need to be exercised [7]. Thus, the presence of process variation poses significant challenges for analysis and optimization of circuits.

Variability in device parameters impacts both timing and power characteristics of digital circuits. An especially important impact is on leakage power consumption. Leakage power growth presents a major challenge to digital circuit design in nanometer scale silicon technologies. The contribution of leakage to total power consumption is nearly one half in 45nm CMOS [81][82]. Leakage power increase is due to both increased subthreshold and gate leakage currents. Subthreshold leakage power increase is due to the reduction of threshold voltage required for frequency scaling and because of the strong exponential dependence of subthreshold leakage on Vth. The gate leakage is due to tunneling current through gate oxide. The aggressive scaling of oxide thickness to control short channel effects, leads to significant gate leakage. Among the two leakage components, the subthreshold leakage is most sensitive to the parametric variation. Leakage depends strongly (exponentially) on Vth and channel length. This strong dependence causes a large spread in leakage current in the presence of process variations: a 30% variation in the channel length can lead to 20X variation in leakage current [3].

This dissertation also looks ahead into the post-silicon era and studies a question of how circuits can be designed in emerging technologies. Currently, the number of options for the technology to replace CMOS is large. Extensive research is being conducted in carbon nanotube (CNT) based electronics, and molecular and quantum devices [61] as replacements for CMOS electronics at the end of scaling. It appears that one of the severe challenges of any replacement technology is extremely high defect density of devices that is orders of magnitude higher than that for CMOS. Thus, the emerging technologies will be characterized not only by high parametric variability but also with a high defect density. Implementing circuits in such high defect densities is a very challenging task.

This thesis contributes towards solving several problems related to design under high parametric variability and high defectivity. These contributions can be broadly classified as solving a set of optimization problems in the synthesis of circuits and developing new analysis techniques. More specifically we address the following problems:

1. **Problem-1:** *Whether the process variation and the information about input vector probability can be used during technology mapping to optimize the yield of the circuit limited by the leakage under timing constraints?*
2. **Problem-2:** *How we can enable efficient and accurate statistical static timing analysis taking into account slope information for arbitrary distributions of process parameters including spatial correlation?*
3. **Problem-3***: What is the effectiveness of the coding techniques employing simple decoders for error protection in high defect density nanofabrics? Furthermore, how the default area overhead of coding schemes can be reduced by exploiting properties of the Boolean circuit?*

1.1 CONTRIBUTIONS TO TECHNOLOGY MAPPING

Technology mapping (TM) is an important step in the logic synthesis. TM is an optimization problem where one seeks to map a Boolean graph onto gates from a given technology. The objective is to minimize the costs such as leakage power under timing constraint. The prior work in TM [8][9][11][12][25][35][36][37] has ignored the process variations and modeled the cost parameters, such as leakage and delay, deterministically. The decision-making of the technology mapping phase must be made aware of its impact on circuit variability. Here we describe the first rigorous work on technology mapping which can explicitly take the process variability into account while mapping a Boolean network to a given technology library. Additionally, we explore the possibility of using TM to reduce leakage power consumption. We have investigated the potential for such power reductions by developing a family of new mapping algorithms.

Specifically, in this dissertation we propose extensions of dynamic programming techniques to do TM taking the impact of process variation and input state probability information in account. In particular, we propose a statistical TM algorithm for maximization of the leakage power yield, under the delay constraint which is also probabilistic. The results indicate that a statistical technology mapping algorithm can produce mappings with substantially reduced power consumption at the same power and timing yield levels compared to an algorithm which does not take into account the distribution of process parameters and uses corner-based value. The details of this work are presented in Chapter 2, and were published in [58].

Minimization of leakage power consumption is an important theme of this dissertation. When attempting to minimize runtime leakage power consumption, one realizes that leakage is strongly dependent on the input state. Leakage is highly sensitive to the value of input vector: for single gates the difference between the max and min

leakage can be up to 10X [29]. For larger circuits the range is smaller but still remains substantial. At design time only statistics on input vectors can be used, however, and the state can be described as a random variable. We have taken into account the impact of primary input state probabilities to minimize the expected leakage, or leakage at a high percentile point under deterministic delay constraint. Compared to the above formulation of TM the new algorithm uses a gain-based formulation for achieving run-time reduction and for handling nearly continuously sized libraries. We show how the dependency of leakage on the output capacitance in gain-based setting can be dealt with, something that the prior work in gain-based synthesis [8][26] could not do. The details are presented in Chapter 3 and were published in [59].

Next, we argue that using the point state probability values for primary input signals in some cases can be impractical and unrepresentative. Instead, we propose to use ranges of probabilities. An experiment with a microprocessor simulator running a variety of applications strongly indicates that the primary input probability is best described as belonging to an interval. We introduce the notion of proxy objective function to optimize the leakage making it less sensitive to specific instance probabilities. We define proxy leakage metric to be the sum of worst mean leakages of all the circuit gates under all feasible workloads. We formulate and develop a new technology mapping algorithm to minimize proxy mean leakage in the framework of recently introduced robust dynamic programming. The uncertainty is modeled using discrete Markov Decision Processes with uncertain transition probability. We develop an efficient solution whose complexity is linear in the circuit size. The effectiveness of the proxy leakage metric was tested by running the gain-based algorithm at point probability e.g. middle interval point. Since depending on the application phase the actual probability during the run-time may shift to some value other than this point probability, we compared the result using different test

point probabilities chosen within the interval as the test probabilities. The average improvement across different benchmarks can be up to 9% and max improvement among different benchmarks can be 16% for certain test probabilities. We also propose an algorithm for computing the interval state probability for the intermediate signals based on the information of the state probabilities of the primary inputs. The details are described in Chapter 4.

1.2 CONTRIBUTIONS TO TIMING ANALYSIS

Static timing analysis is concerned with verifying full-chip timing behavior. Traditionally, timing conformance of the circuit to requirements was checked using deterministic static timing analysis at all the process corners. Statistical static timing analysis (SSTA) was recently introduced. It models the process parameters and delay values as explicit random variables. It enables one to make yield predictions which can not be done in the process corner based approach. However, most of the previous work [42][41][49][51][52] is unable to correctly capture the dependence of delay on the output slope of the previous stage. It either uses worst case slope propagation or latest arrival time slope propagation in a node based approach [42]. These approaches of worst case or latest arriving time slope propagations have been shown to yield less accurate results for deterministic static timing analysis in [27].

We have proposed an efficient path based approach which can correctly propagate the slope information within the SSTA algorithm. The proposed algorithm can be scaled up to one million paths. We have also motivated the feasibility of a path-based algorithm in timing analysis. We provide empirical evidence that the number of paths grows as $n^{1.6}$, which can be handled in reasonable run-time. The work is described in Chapter 6 and was published in [60].

1.3 CONTRIBUTIONS TO CODING OF BOOLEAN FUNCTIONS FOR FAULT-TOLERANCE

In the context of high defect densities which are likely to occur in future nanotechnology, we propose a design methodology for building reliable computational elements using devices manufactured in these technologies. The fundamental strategy of fault-tolerance is not reconfiguration, but low-level protection of individual Boolean functions through the use of efficient coding. The Boolean functions are represented in terms of ROMs or truth tables, and a novel version of the Hamming code is used to protect the functions. The proposed coding strategy exploits for the first time the structure of Boolean logic networks to produce better codes with respect to the area overhead.

The proposed computational architecture is based on a heterogeneous CMOS - carbon nanotube fabric. It seems certain that the early practical uses of CNT-based electronics will be built on top of the heterogeneous CMOS-CNT processes [69]. The enormous economic investments into the CMOS technology and design infrastructure provide a strong impetus to leverage the existing flows as much as possible. Interfacing CMOS and CNT technology is technically feasible; the economic viability of integration is seen by the recent commercialization of non-volatile memory circuits based on CNT integrated with CMOS. Our approach is predicated on optimally combining reliable, albeit low performance, CMOS components with high performance, albeit unreliable, CNT devices. Chapter 5 describes the work that was published in [105].

The novel contributions of this dissertation can be summarized as follows:

1) A new TM algorithm for maximization of yield limited by the leakage power under delay constraints.
2) Extension of gain-based TM algorithm to accurately propagate the load-dependent leakage costs during the synthesis flow.

3) Efficient and compact code construction that utilizes the special structure of Boolean circuits.
4) A fast method for obtaining the delay distribution of a circuit while doing slope propagation and handling spatial correlation.

Chapter 2. Statistical Technology Mapping

We first provide some background for the statistical TM to do leakage optimization under the influence of process variation.

2.1 Need For Variability Aware Synthesis

It is projected that at the 65nm node, leakage power will account for 45% of total power of the circuit [3]. Because of its exponential dependence on the highly varying transistor channel length and threshold voltage, leakage is greatly affected by variability. *Parametric yield* is the probability of meeting the specified timing or power constraints, given the variability of process parameters. Parametric yield loss due to leakage power constraints is a well-known problem [28]. Thus, power optimization that ignores leakage variance is highly suboptimal. Post-synthesis techniques for leakage reduction based on the use of dual-Vth cells, dual-gate length cells, and stacked devices have been very successful [4][5][6]. Yet, a significant untapped potential exists to reduce circuit leakage at the synthesis phase, specifically, using technology mapping.

Leakage power is an exponential function of two highly-variable process parameters (L_{gate} and V_{th}). The exponential dependence causes a large spread in leakage current in the presence of process variations. It has been demonstrated that a 1.3X variation in the effective channel length could potentially lead to 20X variation in leakage current [3]. Experimental studies also show that in a rich library with dual-Vth cells, the sensitivity of leakage to the various process parameters are widely different, and the leakage variances differ markedly from cell to cell. The deterministic TM algorithm completely ignores the differences in cell's leakage variances to guide the optimization. As a result, minimization of the average circuit leakage does not guarantee the optimal

result for leakage at higher quantiles of the distribution. Because much of the parametric yield loss is due to high leakage, what is required is *parametric yield* maximization, that is, the minimization of leakage at higher quantiles.

The potential for leakage minimization at the synthesis level, and the impossibility of separating leakage minimization from parametric yield maximization has been the essential premise of this proposed TM work. This premise justifies propagating the statistical information higher in the synthesis flow and proposing a statistical technology mapping algorithm for leakage power-limited parametric yield maximization under timing yield constraints.

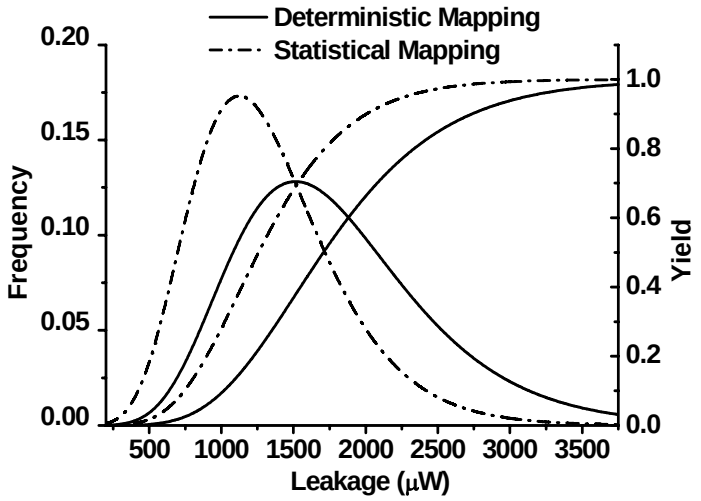

Figure 2.1 The yield and pdf of leakage for C5315 from the statistical and deterministic mappers. The frequency axis denotes the rate of change of yield with leakage constraints. Yield is increased to 80% at a leakage for which the deterministic mapper guaranteed only a 50% yield.

The decision-making of the technology mapping phase must be aware of its impact on circuit variability. This is the first rigorous work on technology mapping which

can explicitly take the process variability into account while mapping a Boolean network to a given technology library.

The results indicate that a statistical technology mapping algorithm can produce mappings with substantially reduced power consumption at the same power and timing yield levels compared to an algorithm which does not take into account the distribution of process parameters and uses corner based value. An example of such improvement in yield is in Figure 2.1 which was produced by the algorithm for one of the benchmark circuits in the ISCAS-85 suit.

Technology mapping is defined as the task of transforming a given Boolean DAG into a network of gates which are taken from a given *technology library*. The mapping has to be done so as to meet some constraints in terms of area, delay or power. The problem that is closely related to TM for parametric yield is that of power minimization under delay constraints. In a conventional setting the problem is solved using dynamic programming, and other effective and well-established techniques [15].

In traditional TM, the costs are determined by the worst-case, or alternatively mean, values of the costs. When the power and delay cost metrics are impacted by variability, and should be both defined probabilistically, the algorithms developed so far fail to find an optimal solution. The reason for such behavior is simple: the deterministic mapping ignores the dependence of the overall cost function on the variance in making the decisions dynamically. Statistical TM explicitly takes this dependence into account. Taking variance into account explicitly is necessary for two reasons. The first one is the significant intra-chip process parameter variation. The second reason is the mistracking of cell power and delay responses due to the differences in cell sensitivities to chip-to-chip variation. One can show that if the leakage or delay sensitivity to each of the global process variation is identical for all cells, then the deterministic algorithm can find

mappings that are as good as those found by a statistical mapping algorithm. In practice, however, there is marked difference in the cell sensitivities.

The actual analysis of the mapping produced by the statistical technology mapper indeed shows its superiority in maximizing yield, or alternatively, in minimizing leakage power at the fixed yield level compared to the deterministic mapper.

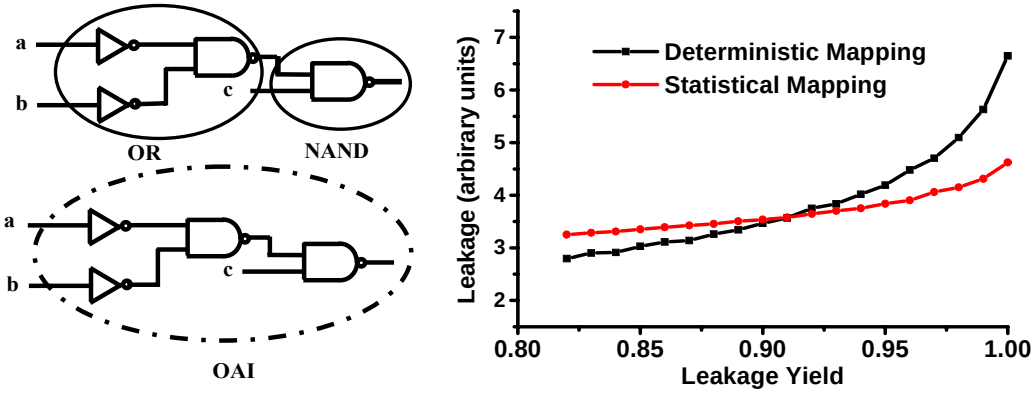

Figure 2.2: A statistical mapping (mapping using OR and AND gates) for yield would result in a higher mean leakage but lower leakage at higher yields (above 95%) compared to deterministic mapping (shown with OAI gate).

Using a well-characterized statistical technology library for a 70nm CMOS technology, the two structurally different mappings of the same subject graph are produced by the two mappers. While the deterministic TM results in the smaller mean power, the statistical solution has higher parametric yield. A deterministic mapper, using the nominal cost values, will reject the first mapping as inferior in terms of both leakage and delay. However, a statistical mapper will recognize the fact that the first mapping minimizes leakage at the 95% yield, Figure 2.2. The proposed algorithm formalizes these trade-offs to enable optimization of large circuits.

The algorithm for deterministic TM is fundamentally based on dynamic programming. The deterministic cost metrics for area, delay, and power, because of the additivity, can all be computed recursively. This enables the principle of optimality [33], and leads to efficient dynamic programming-based implementations. On the other hand, TM for parametric yield has to utilize cost metrics formulated probabilistically, the yield functions, which are intrinsically non-additive. As a result, there can be no simple recursive equations that would permit making optimal decisions at the intermediate stages of the dynamic algorithm while not knowing the cost of the unmapped part of the network.

TM for parametric yield has to essentially be formulated probabilistically. Thus, it makes a radical departure from the earlier work in logic synthesis, and introduces a number of novel challenges. The algorithmic challenges can be summarized as follows: (1) construction of data structures encoding the delay-power tradeoff *Pareto points* capturing the probability distribution; (2) efficient heuristic strategies enabling efficient sequential decision making (pruning) of Pareto points enabling trade-off between quality of mapping and memory complexity; and (3) definition of the analogous operations for *merging* and *addition* of Pareto points used for cost propagation in the deterministic TM [11].

As far as we know, that is the first work on statistical technology mapping. While much work has been done recently to develop statistical timing analysis methods [18], very little work has been done to account for variability in circuit optimization for leakage power minimization via sizing and dual threshold voltage assignment [19][20]. In [19], an iterative TILOS-like approach is extended to rely on statistical sensitivities.

The proposed statistical technology mapping for parametric yield finds a circuit mapping such that the yield at a required power objective is maximized while meeting the required timing constraints at a pre-specified yield level.

The problem of parametric yield maximization can be shown to be equivalent to that of minimizing the quantile function of power under the constraint on the quantile of circuit delay. Fundamentally, the operation of the algorithm is akin to technology mapping algorithm for area under delay constraints [15]. In the algorithm we represent the alternative mappings using power-delay Pareto sets. During the post-order traversal of the canonical circuit DAG, partial coverings are generated and power-delay curves are propagated. However, unlike the deterministic mapper, additional probability distribution information is encoded and propagated. In a statistical framework, the classical principle of optimality does not hold [33] because quantile function is non-additive. We developed a *conditional pruning strategy* to predict, during the intermediate steps of the dynamic algorithm, whether a particular covering has inferior leakage and delay characteristics for all possible coverings of the unmapped circuit. This is an important innovation of our work and is in contrast to previous approaches [16] where decisions are made independent of the yet unmapped part. Probabilistic delay estimation for alternative mappings is based on an efficiently-computable lower bound on the timing quantile cost function. Probabilistic leakage power estimation is also performed analytically. A disutility function that penalizes mappings with high leakage variance is used to guide decision-making under uncertainty towards mappings with lower leakage value at pre-specified percentiles.

The developed pruning strategies enable a direct application of the linear-complexity DAG traversal-based dynamic algorithm used for deterministic technology

mapping [9]. We also exploit the heuristic techniques developed in traditional technology mapping for coping with the NP-hard problem of optimal covering, such as output pin re-ordering [11], alpha-approximate pruning [17], fanout heuristics.

The theoretical complexity of the algorithm is $O(nm^{2a})$, where *n* is circuit size, *m* is the user set parameter for controlling the maximum number of Pareto points that may be stored at each node of the circuit, and a is average number of pins per library gate.

2.3 TECHNOLOGY MAPPING FOR PARAMETRIC YIELD: PROBLEM FORMULATION

Formally, the parametric yield maximization problem of primary interest is:

$$\max \ \ Y(power\_max), \text{ s.t. } Y(delay\_max) \geq \alpha$$

First, this optimization problem is converted to an equivalent problem that is analytically and computationally more attractive. The monotonicity of the yield function permits such a transformation.

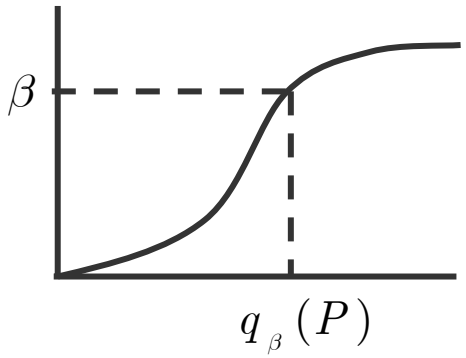

Figure 2.3 Quantile function for the random variable P.

We turn the original optimization problem of maximizing the yield at a fixed upper leakage constraint to the *dual* problem of minimizing the leakage power at a *given yield* α under the same delay constraints. It is possible to retrieve the solution to the

original optimization problem by solving a set of dual problems for a range of α. In the remaining part we focus on the analysis and implementation of the dual problem. The new objective seeks to minimize P_{leak} for a given yield α under a delay constraint $D_{\max}$ for a given yield β. To formally state the problem, the notion of a *quantile* function is introduced.

$$q_\alpha(D) = \min\{t \mid \Pr(D \le t) \ge \alpha\}$$
$$q_\beta(P) = \min\{p \mid \Pr(P \le p) \ge \beta\}$$

Then, the statistical TM for quantile optimization can be expressed as:

$$\text{TM1} : \min q_\beta(P) \quad s.t. \; q_\alpha(D) \le D_{\max}$$

2.4 CELL DELAY AND POWER MODELS

The canonical SIS [30] pin-to-pin delay model is adopted:

$$d(i, C_L) = d_o(i) + \alpha C_L$$

Here, $d_o(i)$ is the intrinsic delay of cell *i*, C_L is the total capacitance loading the output of gate *i*. The amount of leakage current is strongly dependent on the inputs applied to the CMOS gate. It is known, however, that only a small number of states are responsible for the overwhelming portion (e.g. 95%) of the subthreshold leakage current [102]. Using the idea of the dominant state, we can approximate the static power consumption of a gate by:

$$P_{leak} = (1 - \alpha)\sum_i p_{leak,i}\beta_i$$

where $P_{leak,i}$ is the leakage current for a gate in dominant leakage state *i*, β_i is the probability that the gate is in that dominant state, and α is the switching activity factor of the gate. The overall circuit power consumption is computed by summing over all gates.

Library binding for timing and power-limited parametric yield requires superimposing on top of the traditional delay and power models their variational counterparts. Let us assume that there are only two sources of uncertain (variable) process parameters, X and Y. The process variation for each parameter is assumed to be Gaussian, and has an inter-chip and intra-chip component: $\Delta X = \Delta X_l + \Delta X_g$ and $\Delta Y = \Delta Y_g + \Delta Y_l$, where ΔX_l, ΔY_l are the local (intra-chip) components and $\Delta X_{g,}$ ΔY_g are the global (chip-to-chip) components of variation. For compactness, the general ΔX and ΔY notation is used to refer to two primary sources of variability of concern: effective channel length (L_{eff}) and gate-length independent variation of threshold voltage (V_{th}). All the derivations are extendable to additional sources of variation.

Now the cell delay and power properties need to be described statistically. The near-linear dependence of delay on the above process parameters justifies a first-order Taylor expansion of the gate delay function:

$$d_g \cong d_g(L_o, V_{tho}) + (\partial d_g / \partial L)\Delta L + (\partial d_g / \partial V_{th})\Delta V_{th}$$

Under this model, the delay is Gaussian. It was shown in [28], that an empirical leakage power model $P_{leak} = c_o e^{-c_1 L - c_2 V_{th}}$ with constants c_0, c_1 and c_2 can be used to accurately describe the variation in leakage power. Under this model, the leakage power, and hence, total power P is a log normal random variable. Because of the exponential dependence of leakage power on L and V_{th}, the linear approximation is carried out on the log of leakage of each library gate:

$$\begin{aligned} d_i &= d_{i,nom} + d_{i,x}\Delta X + d_{i,y}\Delta Y \\ &= d_{i,nom} + d_{i,x}\Delta X_g + d_{i,y}\Delta Y_g + (d_{i,x}^2 + d_{i,y}^2)^{1/2}\Delta Z_i \\ \log p_i &= p_{i,nom} + p_{i,x}\Delta X + p_{i,y}\Delta Y \\ &= p_{i,nom} + p_{i,x}\Delta X_g + p_{i,y}\Delta Y_g + (p_{i,x}^2 + p_{i,y}^2)^{1/2}\Delta Z_i \end{aligned}$$

The sensitivity coefficients are scaled so that each of random variables can be assumed to be unit normal *N(0,1)*. The local variations ΔX_l and ΔY_l are merged together in a single independent component of variation ΔZ_i, also scaled to unit normal. The impact of independent component variation ΔZ_i on the total leakage distribution of the circuit is negligible [22], so its coefficient can be set to 0.

The *sensitivities* of global components ΔX_g and ΔY_g to leakage are defined as $p_{i,x} / p_{i,nom}$ and $p_{i,y} / p_{i,nom}$ respectively. The statistical delay and leakage models for each cell in the library are characterized and captured in the above canonical form.

2.5 STATISTICAL GRAPH COVERING

Just as the deterministic algorithm [15], the statistical mapper begins by first doing the technology independent optimization. Then, the original circuit netlist is converted to a canonical form, the *subject graph*. The subject graph is a DAG consisting only of base functions, e.g., NAND2 gates and inverters. All the library gates are also converted to canonical forms called *pattern graphs*. Next, using the DAG matching algorithm all possible *matched patterns* of library gates in the subject graph are generated utilizing the DAG matching techniques [15]. After generating the possible matched covering, the deterministic algorithm performs the post-order traversal of the subject graph propagating the power-delay trade-off points through the graph. For the deterministic mapper, a single Pareto point is just a tuple of delay and power whereas in the statistical setting it is a pair of power and delay probability distributions. The propagation of the Pareto points is done by the operations of *adding*, *merging* and the *pruning* of inferior Pareto points [11]. In contrast to the TM in a deterministic setting, the statistical TM algorithm must propagate the information about the variance.

A Pareto point at a subject graph node *n* is associated with a covering at the fan-in cone of *n*. The statistical estimates of the cover's timing and power are computed. Several well-developed statistical timing techniques exist – block based [18][52] path based - [47][31][60]. We incorporated a path-based algorithm in the present implementation.

The statistical operations of merging and adding of the delay component can be seamlessly integrated with the recursive dynamic programming flow of our algorithm. Let the statistical operations of addition and maximum be denoted as $\oplus$ and *max* respectively. We can define the *adding* and *merging* functions as shown below.

$$\mathrm{add}((D_1,P_1),(D_2,P_2)) = (D_1 \oplus D_2, P_1 \oplus P_2)$$
$$\mathrm{merge}((D_1,P_1),(D_2,P_2)) = (\max(D_1,D_2), P_1 \oplus P_2)$$

The basic flow of the DAG covering is similar to its deterministic version [15]. During the post-order traversal at the current node *n* of the subject graph, a new mapping at the fan-in cone of *n* is generated for all possible combinations of Pareto points of the children of the matched patterns at *n*.

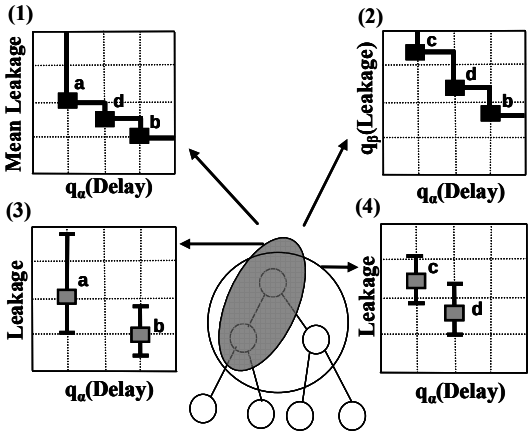

Figure 2.4 Pareto sets (3) and (4) are obtained by adding and merging. Set (1) obtained by the deterministic pruning drops *c* because of its higher mean. But *c* has smaller variance compared to *a* and higher yield. The statistical mapper drops *a*.

If some children nodes of the pattern have multiple fanouts, then before summing up their power distributions, the power is divided by the number of fanouts. This is a standard heuristic used to avoid overestimation of the cost in the Pareto points due to path re-convergence [15].

The primary challenge of statistical TM is that the principle of optimality is not satisfied for the quantile functions. Therefore, the naïve *pruning* strategy of comparing the Pareto points just on the basis of statistical costs (quantile functions) of the corresponding partial mappings will fail, and a more sophisticated method is required.

The algorithm reduces the selection criterion for two Pareto points to an effective procedure based on verification of a collection of inequalities in two variables (section 2.5). The pruning operation in terms of inequalities identifies points that would lead to final mappings with both higher power and delay for all possible mappings of the unmapped part of the circuit. Figure 2.4 compares the pruning of deterministic and statistical mappers for the same set of Pareto points obtained after adding and merging.

The pruning procedure is crucial to scalable memory and time-complexity of the algorithm, since it reduces the number of Pareto points to $O(n_1 + ... + n_i)$, where n_i is the number of Pareto points at *input* i. Without any pruning the number of points grows as $O(n_1 \times n_2 ... \times n_i)$, making the algorithm infeasible [9]. Despite the pruning, a compression strategy is needed to further make the number of Pareto points manageable.

In the domain of deterministic mapping equidistant pruning [9], or pruning based on binning the cost (power or time) at ε length intervals and choosing the best map in each interval has been proposed [15]. More recently, [17] showed that in some cases the error introduced by these compression methods may become unbounded and proposed an alpha-approximation compression algorithm for better results.

In order to provide a trade-off between under-optimality and memory complexity, the alpha-approximate pruning strategy which allows compressing the number of Pareto points needed to be stored exponentially was adopted. The degree of reduction is modulated by a pruning factor α. This method is proved to be within α% of the optimal for the trees and empirically gives good results for DAG [17]. The break points $\lambda_\alpha^i = c_i(1+\alpha)^i$ are instantiated, and the smallest and largest cost (area or power) Pareto points within the interval $[\lambda_\alpha^i,\ \lambda_\alpha^{i+1}]$, where c_i is the smallest cost, are chosen. In the context of the alpha-approximate pruning, the cost metric for power is set up as a disutility function, $E[P_{leak}^2 + P_{leak}]$, where P_{leak} is the random leakage power. Lemma 2.5 establishes that the lower value of the disutility function helps in charactering the set of coverings having lower power quantile values.

1. Start with SIS generated canonical DAG and canonical library patterns.
2. Visit node *n* of the subject graph in topological order.
3. If graph traversal is finished goto 7. Else proceed to 4.
4. Generate all Pareto points at node *n* based on the library patterns matched at *n*.
5. Apply statistical pruning followed by alpha-approximate pruning.
6. Final mapping is recovered by pre-order traversal.
7. Select the lowest cost mapping meting the timing quantile constraints.

Figure 2.5 Pseudo-code for the statistical mapping algorithm.

The algorithm makes a post-order traversal and generates delay-power trade-off curves followed by pruning whenever required. When post order traversal completes, the

final coverings are generated by pre-order traversal starting from the primary outputs in the descending order of logic depth [11].

The complexity of merging, adding and pruning operations is $O(m^a)$, $O(m \times a)$ and $O(m^{2a})$. Here, *m* is the upper bound on the number of Pareto points the algorithm stores and propagates, and *a* is average number of pins per library gate. The overall complexity thus becomes $O(nm^{2a})$, where *n* is the circuit size.

2.6 PATTERN MATCHING BASED ON STATISTICAL COST FUNCTIONS

In this section, the operations of *merging* and *pruning*, the key steps of the algorithm, are formalized and further elucidated. First, the procedure for detecting delay-inferior mappings is described. Stochastic or convex majorization can be used to compare two random delays and find the dominating random variable, which corresponds to the superior mapping [31]. For reasons of extendibility of dynamic programming and computational efficiency, the statistical delay metric should ideally be recursively computable. The original quantile delay cost metric can not be computed recursively. Lemma 2.1 introduces a relaxation of the quantile function by *modified cost objective* based on a lower bound that has a recursive structure.

Note that the using the lower bound is a conservative approach for time-constrained mapping since Pareto points not violating timing constraint are never lost. Although, using the lower bound to guide the comparison of delays of two Pareto points is sub-optimal, experiments show that the bound has good fidelity with respect to the original quantile objective. A similar lower bound has also been used in the context of gate sizing algorithms in [20]. The bound is based on the path-based statistical timing description of the circuit.

Theorem 2.1: *Let D be the random delay of the circuit. Suppose the circuit contains k paths p_i from primary inputs to the primary outputs, D_i is the delay of the i^{th} path, and $d_g \sim N(\mu_g, \sigma_g^2)$ is the gate* g*'s delay. Then,*

$$q_\alpha(D) \geq q'_\alpha(D) \equiv \max_{1 \leq i \leq k} \left(\sum_{g \in p_i} \mu_g + \varphi^{-1}(\alpha) \left(\sum_{g \in p_i} {\sigma_g}^2 \right)^{1/2} \right)$$

Here $\varphi^{-1}()$ is the CDF of the standard normal random variable.

In order to prove theorem 2.1, first we establish some preliminary lemmas

Definition 2.1: A random variable *X* is said to stochastically majorize another random variable *Y* written as $X \geq_{st} Y$ if:

$$\Pr(X \leq t) \leq \Pr(Y \leq t) \ \forall t$$

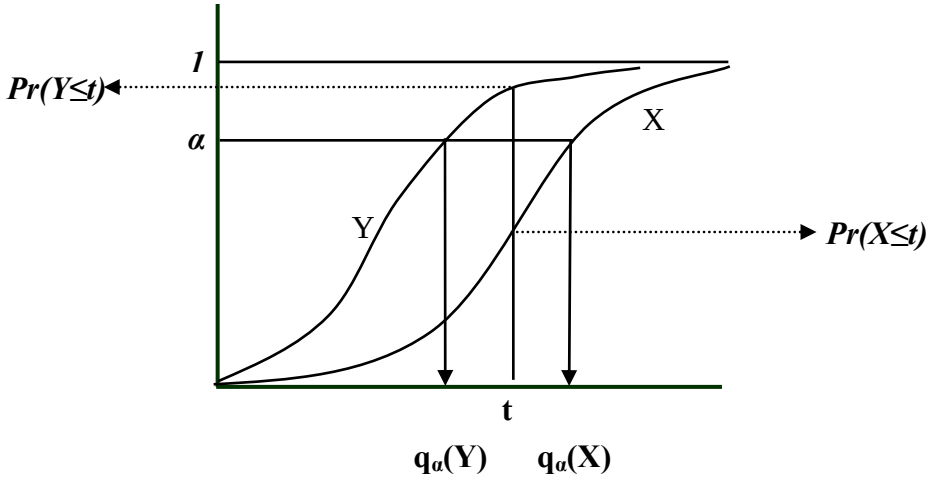

Figure 2.6 Random variable X stochastically majorize random variable Y.

Intuitively it says that in the same number of samples of X and Y, the fraction of values sampled from X higher than any threshold will be more than those sampled from Y. Equivalently the CDF of Y lies completely above X (Figure 2.6).

The stochastic ordering, also orders the quantile value of the random variables as stated in the next lemma

Lemma 2.1: Let X and Y be two random variables such that $X \geq_{st} Y$ then $q_\alpha(X) \geq_{st} q_\alpha(Y)$

Proof: For simplicity we will only prove for the case when X and Y follows a continuous random distribution, although the results is valid in a discrete setting as well. In this case for any random variable Z the quantile function has the property that $\Pr(Z \leq q_\alpha(Z)) = \alpha$.

$$\begin{aligned} \alpha &= \Pr(Y \leq q_\alpha(Y)) \\ &= \Pr(X \leq q_\alpha(X)) \ \leq \Pr(Y \leq q_\alpha(X)) \ [\because \mathrm{X} \geq_{\mathrm{st}} Y] \\ \Rightarrow \ & q_\alpha(Y) \leq q_\alpha(X) \end{aligned}$$

Next lemma establishes stochastic ordering between the delay of a path and the circuit delay.

Lemma 2.2: Let *D* be the random delay of a circuit and D_i be the delay of *i*-th path. The circuit delay D always stochastically majorize the path delay D_i: $D_i \leq_{st} D$.

Proof: Let P be the set denoting the collection of all the paths of the circuit. Then the following holds:

$$D = \max_{j \in P} D_j \leq t \Rightarrow D_i \leq t$$

Hence the probability that *D* is smaller than *t* is always smaller than the probability that D_i is smaller than *t*. Thus from the definition 1, it follows that $D_i \leq_{st} D$.

Now we proceed with the proof of Theorem-2.1

Proof of Theorem-2.1: The delay of i-th path D_i is stochastically dominated by the circuit delay D by lemma-2.2. Hence the α-quantile of D must be more than the α-quantile of the path delay D_i for each path p_i. Now the mean and variance of the delay along path D_i are given below

$$\text{Mean} = \sum_{g \in p_i} \mu_g$$

$$\text{Variance} = \sum_{g \in p_i} \sigma_g^2 + \sum_{g \in p_i} \sum_{\substack{h \in p_i \\ h \neq g}} Cov(d_g, d_h)$$

The aforementioned information combined with the fact that for a normal random variable with mean μ and variance σ^2, the α quantile is $\mu + \phi^{-1}(\alpha)\sigma$. Hence α-quantile of the path delay D_i for each path pi satisfies following under the positive covariance models:

$$q_\alpha(D) \geq q_\alpha(d_i) = \sum_{g \in p_i} \mu_g + \phi^{-1}(\alpha) \sqrt{\sum_{g \in p_i} \sigma_g^2 + \sum_{g \in p_i} \sum_{\substack{h \in p_i \\ h \neq g}} Cov(d_g, d_h)}$$
$$\geq \sum_{g \in p_i} \mu_g + \phi^{-1}(\alpha) \sqrt{\sum_{g \in p_i} \sigma_g^2}$$

Since the previous inequality is valid for each path p_i, it is also valid when we take the max of LHS (left hand side) for all paths p_i. Hence

$$q_\alpha(D) \geq q'_\alpha(D) \equiv \max_{1 \leq i \leq k} \left(\sum_{g \in p_i} \mu_g + \phi^{-1}(\alpha) \sqrt{\sum_{g \in p_i} \sigma_g^2} \right)$$

The *modified delay cost metric* is recursively computable and is therefore used within the algorithm. The optimization strategy is based on minimizing the modified cost function $q'_\alpha(D)$. An additional challenge of statistical TM is the breakdown of the principle of optimality of DP. The algorithm addresses this challenge by introducing a *conditional principle of optimality*. Let M_1 and M_2 be the two Pareto points being compared for delay at a given node *n*. The point M_1 is inferior to M_2 if *for all possible covering of* the unmapped part the circuit delay *D* satisfy $q'_\alpha(D \mid M_1) \geq q'_\alpha(D \mid M_2)$. This condition is framed as a problem of verifying an inequality $q'_\alpha(D_1 + X \mid M_1) \geq q'_\alpha(D_1 + X \mid M_2)$ for any possible path with random delay $X \sim N(\mu, \sigma^2)$ from the node *n* to any of the primary output, D_1 is the delay of the fan-in

cone of n. Within the path-based S-STA approach used in the algorithm, this inequality can be further simplified. Let paths Dp_i^1 and Dp_i^2 be the statistically dominant *p* paths through each of the possible mappings, with distributions $Dp_i^1 \sim N(\mu_{1i}, \sigma_{1i}^2)$ and $Dp_i^2 \sim N(\mu_{2i}, \sigma_{2i}^2)$ respectively with $1 \leq i \leq p$. Then, the inequality reduces to:

$$\max_{1\leq i\leq p}(\mu_{1i} + \mu + \varphi^{-1}(\alpha)(\sigma_{1i}^2 + \sigma^2)^{1/2}) \geq \max_{1\leq i\leq p}(\mu_{2i} + \mu + \varphi^{-1}(\alpha)(\sigma_{2i}^2 + \sigma^2)^{1/2}) \quad (2.1)$$

If this condition is satisfied, then replacing the mapping M_1 by M_2 can never make the modified delay cost function bigger for any choice of feasible mappings in the unmapped part. The above nonlinear problem is solved efficiently using the Lemma 2.3 and Lemma 2.4 below by representing the solution of inequalities (2.1), as a union of intersections of solutions to a basic single-variable inequality (2.2). Based on Lemma 2.3, one can obtain the range of σ for which (2.1) holds. If this range is $[0, \infty)$ then we can say that (5) holds always since μ does not influence the inequality (2.1).

Lemma 2.3: *Let $Dp_i^1 \sim N(\mu_{1i}, \sigma_{1i}^2)$ and $Dp_i^2 \sim N(\mu_{2i}, \sigma_{2i}^2)$ be two sets of normal random variables. Then, for a random variable $N(\mu, \sigma^2)$, the range of σ for which inequality of (4) holds, is, $\bigcup_{1\leq i\leq p} S_i$ and $S_i = \bigcap_{1\leq j\leq p} R_{ij}$ where R_{ij} is the range of σ for which the following basic inequality is satisfied:*

$$\mu_{1i} + \mu + \varphi^{-1}(\alpha)(\sigma_{1i}^2 + \sigma^2)^{1/2} \geq \mu_{2j} + \mu + \varphi^{-1}(\alpha)(\sigma_{2j}^2 + \sigma^2)^{1/2} \quad (2.2)$$

Proof: We observe that (2.1) is satisfied if at least for one i, the following inequality holds

$$(\mu_{1i} + \mu + \varphi^{-1}(\alpha)(\sigma_{1i}^2 + \sigma^2)^{1/2}) \geq \max_{1\leq i\leq p}(\mu_{2i} + \mu + \varphi^{-1}(\alpha)(\sigma_{2i}^2 + \sigma^2)^{1/2})$$

The above inequality is satisfied if and only if for each $1 \leq j \leq p$, the following inequality holds:

$$\mu_{1i} + \mu + \varphi^{-1}(\alpha)(\sigma_{1i}^2 + \sigma^2)^{1/2} \geq \mu_{2j} + \mu + \varphi^{-1}(\alpha)(\sigma_{2j}^2 + \sigma^2)^{1/2}$$

The solution set of the above inequality (for σ) is given as the set R_{ij}. Hence the solution space of the previous inequality is given as $S_i = \bigcap_{1 \leq j \leq p} R_{ij}$. Furthermore $\bigcup_{1 \leq i \leq p} S_i$ is the set of all the σ's for which (2.2) is satisfied. To find R_{ij}, following additional lemma is introduced:

Lemma 2.4: *Let X and Y be two normal random variables with distributions* $N(\mu_1, \sigma_1^2)$ *and* $N(\mu_2, \sigma_2^2)$ *respectively. Suppose Z is another random variable with the distribution* $N(\mu, \sigma^2)$. *Then the range of* μ, σ *for which (2.2) holds*[1] *is given by following set of conditional equations: (*α^* *denotes the positive root of* $\mu_1 + \varphi^{-1}(\alpha)(\sigma_1^2 + x^2)^{1/2} = \mu_2 + \varphi^{-1}(\alpha)(\sigma_2^2 + x^2)^{1/2}$*)*

$$\begin{aligned}
&\text{if } (\sigma_1 \geq \sigma_2 \text{ and } \mu_1 \geq \mu_2) \Rightarrow S = [0,\infty) \times [0,\infty) \\
&\text{if } (\sigma_1 \geq \sigma_2 \text{ and } \varphi^{-1}(\alpha)(\sigma_1 - \sigma_2) < (\mu_2 - \mu_1)) \\
&\qquad \Rightarrow S = \phi \\
&\text{if } (\sigma_1 \geq \sigma_2 \text{ and } \varphi^{-1}(\alpha)(\sigma_1 - \sigma_2) \geq (\mu_2 - \mu_1)) \\
&\qquad \Rightarrow S = [0,\infty) \times [0,\alpha^*) \\
&\text{if } (\sigma_1 \leq \sigma_2 \text{ and } \mu_1 < \mu_2) \Rightarrow S = \phi \\
&\text{if } (\sigma_1 \leq \sigma_2 \text{ and } \varphi^{-1}(\alpha)(\sigma_2 - \sigma_1) \leq (\mu_1 - \mu_2)) \\
&\qquad \Rightarrow S = [0,\infty) \times [0,\infty) \\
&\text{if } (\sigma_1 \leq \sigma_2 \text{ and } \varphi^{-1}(\alpha)(\sigma_2 - \sigma_1) > (\mu_1 - \mu_2)) \\
&\qquad \Rightarrow S = [0,\infty) \times [\alpha^*,\infty)
\end{aligned}$$

Proof: We restate the inequality below:

$$\mu_1 + \mu + \varphi^{-1}(\alpha)(\sigma_1^2 + \sigma^2)^{1/2} \geq \mu_2 + \mu + \varphi^{-1}(\alpha)(\sigma_2^2 + \sigma^2)^{1/2}$$

[1] Note that we dropped the index *i* and *j* for simplicity.

Clearly this inequality holds independent of μ as it cancels on both side. Now in order to determine the range for σ we consider two separate cases:

Case 1: ($\sigma_1 \geq \sigma_2$), we can rewrite the above inequality equivalently as shown below.

$$\varphi^{-1}(\alpha)((\sigma_1^2+\sigma^2)^{1/2}-(\sigma_2^2+\sigma^2)^{1/2}) \geq \mu_2-\mu_1$$

$$\Phi(\sigma)=\frac{\sigma_1^2-\sigma_2^2}{(\sigma_1^2+\sigma^2)^{1/2}+(\sigma_2^2+\sigma^2)^{1/2}} \geq \frac{\mu_2-\mu_1}{\varphi^{-1}(\alpha)}$$

Now we observe that $\Phi(\sigma)$ is a decreasing function of σ. Thus in order for the prior inequality to not hold true for all σ we must have: $\Phi(0) < {\mu_2-\mu_1}\big/{\varphi^{-1}(\alpha)}$ or equivalently $\varphi^{-1}(\alpha)(\sigma_1-\sigma_2) < \mu_2-\mu_1$. Further the inequality will hold true for all σ if $\lim_{\sigma\to\infty}\ \Phi(\sigma) \geq {\mu_2-\mu_1}\big/{\varphi^{-1}(\alpha)}$ or can be simplified to $\mu_1 \geq \mu_2$. If none of the above conditions apply then the range of σ starts from 0 and the end point α^* will be the value of σ for which the equality holds.

Case 2: Now we look at the case $\sigma_2 \geq \sigma_1$. Now we can equivalently write the target inequality as:

$$(\sigma_2^2+\sigma^2)^{1/2}-(\sigma_1^2+\sigma^2)^{1/2} \leq {\mu_1-\mu_2}\big/{\varphi^{-1}(\alpha)}$$

$$\Psi(\sigma)=\frac{\sigma_2^2-\sigma_1^2}{(\sigma_1^2+\sigma^2)^{1/2}+(\sigma_2^2+\sigma^2)^{1/2}} \leq {\mu_1-\mu_2}\big/{\varphi^{-1}(\alpha)}$$

Note that the function $\Psi(\sigma)$ is decreasing in σ. The above stated condition is satisfied for no σ if $\lim_{\sigma\to\infty}\Psi(\sigma) > {\mu_1-\mu_2}\big/{\varphi^{-1}(\alpha)}$ or $\mu_2 > \mu_1$. The inequality is satisfied for all σ if

$$\Psi(0) \leq {\mu_1-\mu_2}\big/{\varphi^{-1}(\alpha)}$$

In other words $\varphi^{-1}(\alpha)(\sigma_2-\sigma_1) \leq \mu_1-\mu_2$. If none of above condition holds then the range of σ will end in ∞ and will begin at α^* where the equality holds.

The leakage costs are distributed log-normally. In contrast to the delay cost metrics which are Gaussian, there are no closed-form techniques for adding the distribution of two log-normal random variables. As a result, the principle of conditional optimality used for ordering alternative mappings with respect to timing, cannot be extended. Thus, the pruning strategy for Pareto points based on their leakage cost metrics uses a strategy of risk-minimization and regularization that penalizes the alternatives with higher mean and variance of power. Similar principle has been employed for delay minimization during gate sizing [20]. The disutility function is defined as:

$$E[P_{leak}^2 + P_{leak}]$$

Here P_{leak} is the random leakage power. It can be shown that minimization of the disutility function tends to minimize the quantile function of power as well, which is the ultimate objective of the optimization (Lemma 2.5).

Lemma 2.5: *Let X be a random variable with E[X] as the expected value. Then*

$$q_\alpha(X) \leq \sqrt{1 + \frac{E[X^2 + X]}{1-\alpha}} - \frac{1}{2}$$

Proof: We use following chebyshev's inequality to prove the above inequality.

$$\Pr(Y > a) \leq E[Y] \Big/ a$$

We substitute $Y = X^2 + X$ and $a = q_\alpha(X)^2 + q_\alpha(X)$ in the above inequality to get the following:

$$\Pr(X^2 + X > q_\alpha(X)^2 + q_\alpha(X)) = \Pr(X > q_\alpha(X)) =$$
$$(1\text{-}\alpha) \leq E[X^2 + X] \Big/ q_\alpha(X)^2 + q_\alpha(X)$$

We can equivalently re-write the above inequality to get an upper bound on $q_\alpha(X)$ we use the fact that $x^2 + x$ is a monotonically increasing function of x.

Lemma 2.6 provides a way to evaluate the disutility function.

Lemma 2.6: *Let M be a mapped circuit containing* $\alpha_1,...,\alpha_n$ *instances of the library gates* $L_1,...,L_n$ *which have leakage powers* $P_1,...,P_n$ *respectively and let P denote the total leakage power value then*

$$E[P^2 + P] = \sum_{\substack{1\le i\le n \\ 1\le j\le n}} \alpha_i\alpha_j E[P_iP_j] + \sum_{1\le i\le n} \alpha_i E[P_i]$$

$$E[P_iP_j] = \exp(p_{i,nom} + p_{j,nom} + \frac{1}{2}((p_{i,x} + p_{j,x})^2 + (p_{i,y} + p_{j,y})^2))$$

$$E[P_i] = \exp(p_{i,nom} + \frac{1}{2}p_{i,x}^2 + \frac{1}{2}p_{i,y}^2)$$

Proof: The power P can be expressed as $\sum_{1\le i\le n} \alpha_i P_i$. Therefore $P^2 + P = \sum_{\substack{1\le i\le n \\ 1\le j\le n}} \alpha_i\alpha_j P_iP_j + \sum_{1\le i\le n} \alpha_i P_i$, thus the first equation of the above stated lemma follows. Now based on the equation in Section 2.3.1 for leakage we have following

$$\begin{aligned}\log p_i &= p_{i,nom} + p_{i,x}\Delta X + p_{i,y}\Delta Y \\ &= p_{i,nom} + p_{i,x}\Delta X_g + p_{i,y}\Delta Y_g + (p_{i,x}^2 + p_{i,y}^2)^{1/2}\Delta Z_i\end{aligned}$$

Assuming the process parameters are normally distributed and neglecting the impact of local process parameter ΔZ_i on leakage distribution, we can say that P_i and P_iP_j have lognormal distribution. For a lognormal random variable $e^{N(\mu,\sigma^2)}$ the mean value is $\exp(\mu + \sigma^2/2)$. Utilizing this result and the fact that ΔX_g, $\Delta \mathrm{Y_g}$ are normally distributed we can get the second and third equalities of lemma 2.6.

This section described the *primary pruning scheme* to reduce the number of Pareto. Additional, alpha-pruning is used to reduce the number of Pareto points further.

2.7 IMPLEMENTATION AND EXPERIMENTS

The technology mapping algorithm TM-PY was implemented in C. It consists of 17000 lines of code, and operates in the standard SIS environment. The experiments have been run on the circuits in the publicly available ISCAS'85 and MCNC benchmark suites. The same technology independent optimized *blif* circuits were used to compare both the deterministic mappings and the statistical mappings. The decomposition of the circuit netlists to the canonical subject graph consisting of only NAND and INVERTER gates was obtained using SIS.

The cell library used to test the performance of the algorithm consists of 16 cells. The library contains both low-V_{th} and high-V_{th} versions of the structurally (logically) different cells. The use of dual-*Vth* cells ensures diverse sensitivity of leakage to process variations between cells. The library was characterized for a 70 nm process using Berkeley Predictive Technology Model. For NMOS (PMOS) transistors, the high threshold voltage is 0.20V (-0.20V) and the low threshold voltage is 0.10V (-0.10V). A richer library could include cells that have a finer Vth allocation granularity, skewed p/n transistor sizing, and even dual L_{gate} gates. It is likely that such a library would enhance the advantages of the statistical mapping (it can be easily done at the cost of higher run-time of the algorithm). The algorithm is based on the standard SIS delay modeling, but can utilize any standard library-based gate delay description including having different rising and falling delays for a gate. Path delays were computed using an internal static timing analyzer. Switching activity and dominant leakage state probabilities were calculated by random simulation. Input probabilities were assumed independent with equal probability of being high or low. The library was also characterized statistically. Two primary sources of variability were considered: effective channel length (L_{eff}) and gate-length independent variation of threshold voltage (V_{th}). These parameters have

significant impact on timing and leakage power. An additive statistical model that decomposes the variability, of both L_{eff} and V_{th}, into the intra-chip and chip-to-chip variability components is used. The variability of L_{eff} is assumed to be 8% of σ / μ, the variability of V_{th} variability is $\sigma / \mu = 7\%$. Inter-chip component of variation is assumed to be 40% of the overall variation.

The experiments to establish the effectiveness of statistical technology mapping for yield were carried out in two modes: (a) maximizing power yield under delay yield constraints; and (b) maximizing timing yield under power yield constraints. The two cases are similar, so we primarily describe the methodology behind the first set of experiments. Since the deterministic algorithm has no way to take into account the probabilistic constraints on timing or leakage power yield, the following search method is used: the deterministic timing constraint is swept across a range of delay values. For each run of the deterministic mapper, a Monte Carlo simulation is performed on the generated mapping and the delay and power values at the specified quantiles are recorded. In this manner, a mapping with the lowest power which satisfies the delay constraint is selected. This mapping is then compared against the result of the statistical mapper. Similar experiments are carried out for leakage constrained statistical mapping.

The experiments indicate that across the benchmarks significant savings in leakage power can be achieved by the statistical mapper as opposed to the deterministic one, Table 2.1. Columns 3-4 of Table 2.1 report the maximum and the average improvements of power at the 99.9% yield, when the delay constraint was also set at the 99.9% yield. The leakage power savings for many benchmarks can be as high as 20-40%, with the average across the benchmarks of 26%, Column 3. The savings also depend on the time constraint. These typical savings range from 3% to 30% with the average of 17% across the benchmarks. In the columns 5-6, the maximum and the average improvement

of delay at 99.9% yield level are analyzed (the power constraint was also set at the 99.9% yield). The delay improvement on average can be as high as 10%. The typical savings across the benchmarks average to 5%.

Figure 2.7 shows the final power-delay curves produced by both the statistical and deterministic mappers for the benchmark circuit alu2. Depending on the precise value of the constraint, the savings in leakage power vary from 8% to 34%. Similar results for delay quantile minimization under leakage quantile constraints are observed, Figure 2.8. As the leakage constraints become tighter, the improvement in delay becomes more significant. We have observed a general trend that the delay savings are higher for the tighter leakage constraints compared to the deterministic mapper, as can be seen in Figure 2.8.

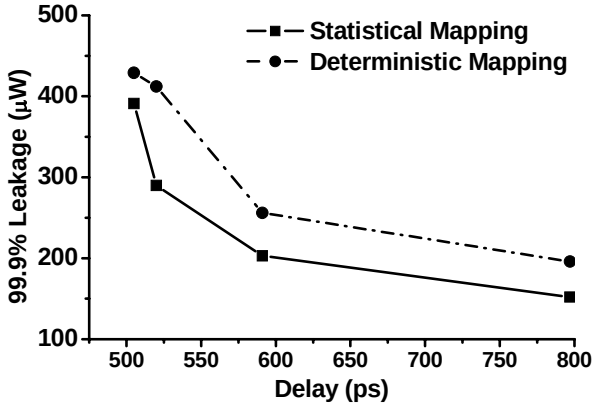

Figure 2.7 Power-delay curves produced by statistical and deterministic mappers for Alu2. Leakage savings are 9% to 30% depending on delay constraint.

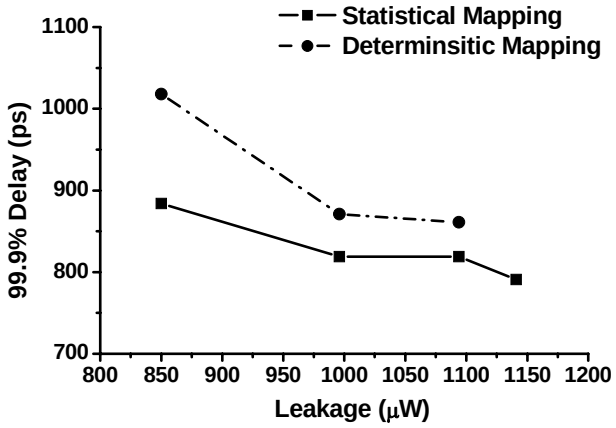

Figure 2.8 Delay-power curves produced by statistical and deterministic mappers for C3540. Deterministic mapper is unable to find a mapping for the tightest delay constraint.

This can be explained as follows: The algorithm is using a disutility function which has the effect of simultaneously minimizing the leakage mean as well as the variance. It has been reported that there is a negative correlation between leakage and delay [22]. Because of the disutility function, the algorithm tends to accumulate the Pareto points having lower leakage quantiles, which, due to negative correlation, also tend to have higher delay. Thus, the algorithm is able to produce high improvements when the leakage constraint is tight or when the delay constraint is loose.

Figure 2.1 depicts the *pdf* and *cdf* of leakage power obtained from the deterministic and statistical mappers by imposing a fixed delay constraint at 99.9% quantile for circuit C5315. For the entire range of leakage values, the parametric yield is significantly higher for the circuit produced by the statistical mapper. For example, the statistical mapper leads to a 80% yield at the leakage value for which the deterministic mapper guaranteed only a 50% yield.

Table 2.1 The experimental results for delay and leakage constrained mappings at 99.9% yield.

Circuit	Size	Leakage Power		Delay Savings (%)		Run time (sec)
		Max	Average	Max	Average	
9symml	311	23	19	13	6	134
Alu2	565	34	21	15	8	510
C1355	706	11	5	4	2	371
C499	711	50	47	6	3	1560
C1908	1167	27	10	9	5	2980
C3540	1942	21	13	17	9	2029
C6288	2603	10	6	2	1	230
C5315	3448	38	18	14	6	2940
Average	1718	26	17	10	5	1344

2.6 CONCLUSION

We presented the first synthesis algorithm that performs maximization of timing and power-limited parametric yield. The algorithm implements a statistical version of technology mapping for power under delay constraints. Across the benchmarks the algorithm achieves significant reductions in leakage power of up to 26%. This is the first practical demonstration of the possibility of rigorous statistical synthesis in the presence of variability. The future work will aim at enhancing the capacity and improving the run-time of the algorithm.

Chapter 3. Gain based TM for minimum run-time leakage

3.1 MOTIVATION

Apart from the process variations, there is large leakage power variability due to dependence of leakage on the input state. Under the scenario of uncertain state information, we developed technology mapping algorithm for leakage minimization using the efficient gain-based setting [8]. Gain-based TM was introduced in [8] and has been essential for solving the problem of timing closure [14]. Gain-based techniques are attractive because of their superior computational efficiency for rich libraries. Technology mapping for leakage has been for the first time investigated in [11] using a load-binning approach. The solution of [11] relies on a heuristic treatment of the load dependency, which would lead to sub-optimal results in the case of a continuously sized gate library. It is also likely to incur high run-time penalty because of the need to discretize gates in such a library.

The problem of TM for leakage in a gain-based setting has not been studied previously. The failure of existing gain-based algorithms which are based on implicit area minimization to find a mapping with minimum leakage necessitates the development of such a technique.

One essential contribution is that the proposed algorithm is formulated to find a mapping that minimizes leakage at an arbitrary quantile of its distribution. Leakage is highly sensitive to the value of input vector: for single gates the difference between the max and min leakage can be up to 10X [29]. For larger circuits the range is smaller but still remains substantial. While average (mean) power determines average battery life, maximum (peak) power directly influences the cooling requirements and the packaging costs. The cost of a cooling solution depends on the maximum power, and increases

dramatically beyond a certain point: a 15% increase in power, can lead to a 3.5X increase in the cost of a cooling solution [10].

Both average and maximum runtime leakage power can be reduced by exploiting the leakage state dependency and the unequal state probabilities and leakage variances. It was observed that a 62% reduction of power is possible if the assignment of transistors in a gate to higher Vth is done taking into account the probability of a gate being in a specific state: for gates with the skewed state probability, only transistors leaking in that state will be set to high Vth, minimizing leakage with minimal delay penalty [13]. While the fundamental observation is correct, the work of [13] is based on the simplifying assumption that the state probabilities are skewed to either 0 or 1, which enables a rather straightforward algorithm for runtime leakage minimization.

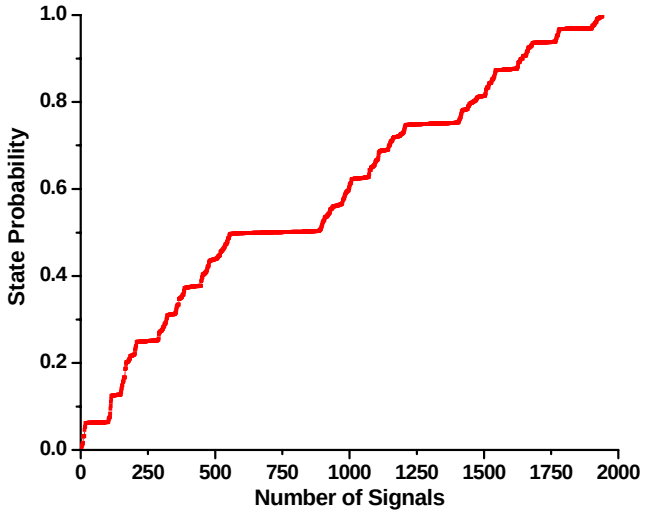

Figure 3.1 The distribution of state probabilities in the signals of a combinational circuit C3540 wherein the primary input probabilities are set to 0.5. The distribution appears to be fairly uniform, except for a cluster around 0.5.

Extensive experiments on general combinational circuits do not validate this assumption (Figure 3.1). Thus, a general algorithmic approach to input state probability-aware

optimization is required and it has been developed in the context of TM. Gain-based TM (GB-TM) algorithms utilize the logic-effort delay models [34], in which delay is composed of two parts. A parasitic delay p and an effort delay part which is directly proportional to the gain $g\ (= C_{out}\,/\,C_{in})$, C_{out} is the output load and C_{in} is the input capacitance. For a fixed value of gain, gate delays are fixed, and minimum delay mapping can be easily performed. Several gain-based TM algorithms fundamentally rely on the monotonic relationship between area and delay which is enforced implicitly via the notion of gain. A simplified approach to target area-delay tradeoff using GB-TM is presented in [8]. A *pivotal gate* is selected and is assigned some global gain value. Then, the gain of all the other library gates can be adjusted so that they have the same effort delay as the pivotal gate [8]. Because of the monotonic one-to-one relationship between delay and via gain, higher gain leads to a higher delay and lower area mapping. Optimal area-delay Pareto curves are obtained by sweeping across the range of global gain values.

However, this technique fails in trying to find a leakage-minimum mapping. The basic reason is that while for area there is a one-to-one mapping between a gain value and area, there is a one-to-many mapping between gain and leakage. For instance, all assignments of high/low threshold voltages to transistors in a gate will be characterized by the same logical effort and gain. A strategy that minimizes delay for a fixed gain in a rich library with mixed-Vt gates would always lead to a mapping with all low Vt. A mapping containing mixed-Vt gates would not be picked, even if its delay is only marginally larger. Because of the exponential delay-leakage tradeoff via Vt, the resulting leakage for a gate may be orders of magnitude higher than minimally possible. Sweeping the global gain value in small steps does not help, since the min-delay and max-leakage solution is always identified. We have experimentally confirmed this by finding circuit mappings that are uniformly better in leakage (Figure 3.2).

A solution to this problem within the gain-based TM framework is for each gain value to propagate a set of competitive delay-leakage pairs. How can this be done efficiently, i.e., without the knowledge of output capacitance? Relying on gain-based delay models removes the problem of delay versus output capacitance dependency. To enable accurate estimation for leakage, we propose a linear model of leakage in terms of the output capacitance of a match. This result is a key enabler in capturing the dependence of leakage on the output capacitance without resorting to a binning based approach. Instead, we propagate the leakage/load slope, and use the value of the slope itself to compare the quality of two Pareto points in terms of leakage.

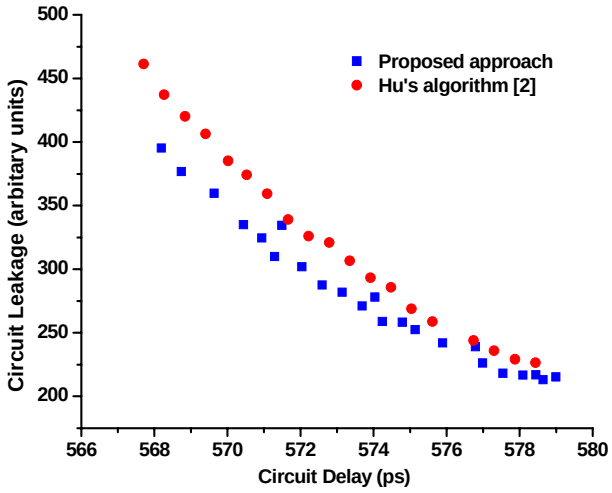

Figure 3.2 The algorithm in [8] is unable to find the leakage-optimal mappings even with a very fine-grained sweep of the global gain (2.11 to 2.30 in steps of 0.01) for the benchmark C1908. Leakage savings 4.8% to 9.6% with no delay penalty.

We also show that during the dynamic programming flow of the gain-based TM, the partial mappings associated with any fanin-cone will have leakage directly proportional to the output capacitance. We provided results that allow us to compute the

slope of a fanin cone as a linear function of the slope of the matched gate at the root node of the fanin cone and the slopes of the fanin mapping of the matched gate.

3.2 ALGORITHM FOR GAIN-BASED TM

In what follows we develop the necessary models and algorithms for gain-based TM. Two algorithms are presented. The first minimizes leakage by taking the leakage cost to be the leakage of the dominant state. The second algorithm extends this approach to minimizing leakage at a given percentile value.

3.2.1 Cell Modeling

A continuous library is assumed and a logic effort model of delay [34] is adopted. In the gain based delay model, the delay of each gate is characterized as $d = g_i h_i + p_i$, where $g_i = C_{out} / C_{in}$ is the electrical effort, h_i is the logical effort and p_i is the parasitic delay. Logical effort theory [34] states that the delay of a path is optimized if the effort delay $g_i h_i$ is balanced along the path. One way to achieve this is to define *a global gain value G* for the inverter or some other fixed library cell which we refer to as *pivotal gate* and then determine the gain value of all the cells according to:

$$g = \frac{Gh_{inv}}{h}$$

Leakage estimation during technology mapping is difficult because cell widths, and thus leakage, are not known until the full circuit is mapped. A novel contribution of this work is the extension of the logical effort based model to leakage. The library consists of *N* cells. Let the delay, capacitance, dominant leakage, width and gain of gate *i* be denoted as $t_i, C_{in}, leak_i, W_i, g_i$ respectively. The cells in the library were characterized using SPICE to obtain the following information for cell *i*: (h_i, p_i, r_i, q_i are constants obtained by characterization)

$$t_i = h_i g_i + p_i = h_i \frac{C_{out}}{C_{in}} + p_i, C_{in,i_i} = r_i W_i, leak_i = q_i W_i$$

After the global gain *G* is decided, the delay of each cell is fixed and the delay of cell *i* is given by $T_i^G = Gh_{inv} + p_i$. If the output capacitance is C_{out}, the leakage of cell *i* can be estimated in terms of the output capacitance as:

$$T_i^G = \frac{C_{out}}{C_{in}} h_i + p_i, \frac{T_i^G - p_i}{h_i} = \frac{C_{out}}{C_{in}}$$
$$C_{in} = C_{out}\left(\frac{h_i}{T_i^G - p_i}\right), W_i = C_{out}\left(\frac{h_i}{r_i(T_i^G - p_i)}\right)$$
$$leak_i = C_{out}\left(\frac{q_i h_i}{r_i(T_i^G - p_i)}\right) = L_i C_{out}$$

Note that now leakage has been expressed as a linear function of output capacitance, a mapping algorithm can be developed which takes the aforementioned library characteristics as input parameters.

3.2.2 Using Dominant-Leakage State as Cost

The initial phase of the gain based algorithm consists of steps which are identical to the binning based approach [15]. Technology independent optimization is followed by conversion of the Boolean netlist to a canonical DAG subject graph *G(V, E)* consisting of only two primitive gates e.g. NAND and INVERTER. Afterwards, all possible matches rooted at each of the node of the subject graph are identified. Since we use a true DAG covering algorithm, matches may include multi-fanout vertices internally.

Dynamic programming is applied in the next phase. It recursively propagates the leakage-delay cost trade-off Pareto pairs for the partial mappings generated at fanin cones of the intermediate nodes of *G*. For a pattern *m* matched at the node *v[i*], we use the merge operation on all possible combinations of Pareto points at the input pins of *m*. Afterwards, the add operation is applied to combine the costs of resultant mappings to *m*. Pruning is subsequently employed to eliminate sub-optimal Pareto points. If the number

of Pareto points in this step still exceeds a predefined threshold, we apply alpha-approximate pruning [17], which results in an exponential decrease in the number of points as a function of the percentage of sub-optimality introduced. In the final stage, we select a Pareto point which has the lowest leakage cost under the target delay from each of the primary outputs. Backward propagation is then performed by visiting the subject graph in a pre-order traversal starting from the library pattern associated with the selected Pareto points at the primary outputs and propagating the slack available to the fanins of the associated pattern.

We show that the leakage of the partial mapping at the fanin cone of any node is a linear function of the output capacitance. Each Pareto point consists of a (d,l) tuple. The element *d* denotes the delay of the partial mapping and *l* is the slope of the leakage with respect to the output capacitance. Thus if the output capacitance is $C_{out,}$ then the leakage value of the partial mapping corresponding to the Pareto point is lC_{out}. Note that C_{out} is a quantity that is unknown in the gain based framework until we have fully mapped the circuit. However for pruning purposes it suffices to have only the slope information. We illustrate the pseudo-code for Pareto point merging and adding for dominant leakage based GB-TM in Figure 3. Let *M* be a pattern match rooted at a subject graph vertex $v[k]$. Let *M* correspond to the library cell c and let *T1* and *T2* denote the partial mapping at the fanins of *M* corresponding to the Pareto point (d_1,l_1) and (d_2,l_2). A new partial mapping with the delay d and leakage slope l is then obtained by taking *T1* and *T2* and adding it to the match *M*. The computation of these values is described below:

$$d = \max(d_1, d_2) + T_c^G$$

The leakage of this combined partial mapping is estimated by observing that the input capacitance of *M* is C_{out} / g_c. Thus the total leakage is:

$$leak = l_1 \frac{C_{out}}{g_c} + l_2 \frac{C_{out}}{g_c} + C_{out} L_c = (\frac{l_1 + l_2}{g_c} + L_c) C_{out}$$

Thus, the slope *l* can be written as:

$$l = \frac{l_1 + l_2}{g_c} + L_c$$

Merge_Add

1. **Define** where is the Pareto set at the *i*-th input pin of
2. Sort the delay components of all the Pareto points in S. Let the delays be
3. For each
   a. Find in each the Pareto point with the lowest leakage slope value under the restriction that the delay component is smaller than. If no such exists then set to be NULL.
   b. Combine the picked up Pareto points's with the match using (1) to get the Pareto point
   c.
4. **Return** which is the set of all the final Pareto points generated.

Figure 3.3 Algorithm to reduce complexity of the pruning operation from $O(p_1 ... p_n)$ to $O(p_1 + ... + p_n)$.

The merging and adding operations can be generalized to the case when the match *M* has *n* inputs and corresponding fanin graphs $t_1,...,t_n$ have associated Pareto tuples $(d_1,l_1),...,(d_n,l_n)$ respectively. The delay, leakage-slope costs (d,l) for the partial mapping obtained by combining $t_1,...,t_n$ and the match *M* can be obtained as:

$$\begin{aligned} d &= \max(d_1,...,d_n) + T_c^G \\ l &= \frac{l_1 + ... + l_n}{g_c} + L_c \end{aligned} \qquad (3.1)$$

In general, we will have multiple Pareto points at the fanins of a match. The operations of merging and pruning can now be generalized to sets of Pareto points. Let match *M* have *n* inputs which are fanouts of subject graph nodes $v_1,...,v_n$. Each of these $v_i's$ has an associated Pareto set $S_1,...,S_n$ which has been assumed to be already

computed since we proceed in a topologically sorted order in the DP framework. A naïve way to merge all the Pareto points would be to choose all possible combinations of *n* Pareto points from each of $S_1,\ldots,S_n$ and merge them using (3.1).

Prune ($(d_1,l_1),\ldots(d_n,l_n)$)

1. Sort the original Pareto set with respect to the delay components in a decreasing order of delay to obtain $(d_1^{'},l_1^{'}),\ldots,(d_n^{'},l_n^{'})$.
2. Initialize:
   a. After_prune_set = $\{(d_n^{'},l_n^{'})\}$
   b. current = (n-1)
3. Let $(d_k^{'},l_k^{'})$ be the Pareto point with the lowest index k in After_prune_set
4. if ($l_{current}^{'} \geq l_k^{'}$)
   prune $(d_{current}^{'},l_{current}^{'})$
   else
   insert $(d_{current}^{'},l_{current}^{'})$ in After_prune_set
5. current = current – 1

Figure 3.4 Algorithm to reduce complexity of pruning Pareto points generated after Merge_Add operation from $O(n^2)$ to $O(n\log(n))$.

However, this has the complexity of $O(p_1\ldots p_n)$ where $p_i = |S_i|$. We can reduce the complexity to $O(p_1+\cdots+p_n)$ by employing the approach presented in Figure 3.3.

For a fixed *M* the Pareto points returned by the Merge_Add function will be co-optimal. However the presence of multiple matches at any subject node *v[i]* will give rise to a combined set of Pareto points, some of which may be redundant. Hence we need a pruning strategy to remove them. Let the Pareto points generated be $(d_1,l_1),\ldots(d_n,l_n)$. A simple strategy is to do pair wise comparison between each pairs (d_i,l_i) and (d_j,l_j) and prune the first point if it has both higher delay and leakage slope. This has the complexity

of $O(n^2)$ which can be reduced to $O(n\log(n))$ by using the approach presented in Figure 3.4.

The Pareto points are propagated during the post-order traversal shown in Figure 3.5. We introduce a dummy sink gate with delay *0* and leakage slope *0* which has its inputs as all the primary outputs of the circuit.

Based on the delay constraint, the mapped circuit is recovered during a pre-order traversal of the subject graph. During the pre-order traversal, the arrival time constraints are propagated backwards starting from primary outputs. The library patterns are bound to specific subject graph nodes so that we cover the full subject graph. We always choose Pareto points which satisfy the arrival time constraints with lowest value of leakage slope. For example suppose we encounter a Pareto point *M* which corresponds to pattern *c* which has delay *d*. Further suppose that *M* has an arrival time constraint *a*. The arrival time constraints at *M*'s fanins become *a-d*. We call *a-d* as the arrival time imposed by *M*. At each fanin of *c*, we pick up a Pareto point with lowest leakage slope which satisfies the arrival time *constraint imposed by M*. In case of multiple primary outputs the quality of the mapped circuit depends on the order in which we process the primary outputs. It is usually most beneficial to perform pre-order traversal in the order of the most critical (in terms of available slack) to least critical primary output [11].

In case of multiple primary outputs we may run into a scenario where during the post order traversal some fanin of the pattern associated with the current Pareto point *M* may be already mapped. Suppose one such fanin cone is rooted at subject graph node *v*. In this case, we check if the currently mapped Pareto point *P* at *v* meets the arrival time constraint imposed by *M*. If this occurs, we don't change the mapped pattern assigned at that fanin. Otherwise, we replace *P* by the best leakage slope Pareto point *Q* rooted at *v* which meets the new arrival time constraint.

Forward_Pass (Subject Graph $G(V,E)$)

1. Visit the subject graph nodes in topological order
2. For current node *v[i]*.
3. For all match m rooted at *v[i]* and associated library gate pattern *c*
   a. **Set** $S_m =$ **Merge_Add**(m,c)
   b. **Set** $S = \bigcup S_m \forall match\ m$
   c. **Set** $S' =$ **Prune**(S)
   d. $v[i].P = S'$

Figure 3.5 Algorithm for post-order traversal of subject graph.

This is accomplished in two steps: removal of the previous pattern *p* bound at *v* and recursive update of fanin cones of *p*. Potentially the mappings at the fanin cones of *p* can be improved by the removal of the arrival time imposed by *P*. Recursive propagation of the arrival time constraint is then performed at the fanin cones of the children of the patterns associated with *Q*.

3.3 TM FOR QUANTILE LEAKAGE OPTIMIZATION

The switching probabilities of the input vector state are captured by specifying the probabilities of each of its primary input being 1. Because of dependence of leakage on the input state, leakage becomes a random variable. The proposed technique for gain-based technology mapping minimizes the value of leakage power at a certain quantile α.

$$\min_{d_{circuit} \le d_{given}} q_\alpha(leakage)$$

In order to map a circuit such that leakage at any quantile value is minimized (due to input state uncertainty), we would like to characterize the leakage of library gates for

each input vector instead of the worst case leakage. Let the library gate has *i* have x_i inputs. Thus, it has $y_i = 2^{x_i}$ input states. In an input state *e*, let the leakage be characterized as $leak_i^e = q_i^e W_i$ where, W_i is the width of gate *i*.

We evaluate input state probability at each of the intermediate signals of the subject graph by random simulation. The random simulation is done by generating the primary inputs in each of the iterations of the Monte Carlo simulator based on the primary input signal probabilities. We associate a function with the leakage random variable *L* that penalizes for high leakage variance defined as: $Penalty(L) = E[L] + \lambda\sqrt{Var(L)}$.

The *penalty parameter* λ allows for tuning of the penalty accorded to the variance. In general we run through various sweeps of the algorithm for different value of λ. Similar to the dominant case leakage, it can be shown that this penalty function is also a linear function of the output capacitance. We illustrate its computation assuming that the gate *i* is facing output capacitance C_{out}. Let the input states are $e_1,...,e_n$ with probabilities $p_1,..., p_n$. According to the equations in Section 3.1.1 derived for the leakage in terms of the output capacitance, we deduce that leakage in the state e_j is given by:

$$leak_i^{e_j} = C_{out}\left(\frac{q_i^{e_j} h_i}{r_i(T_i^G - p_i)}\right) = C_{out} L_i^{e_j}$$

Hence, the penalty function can be computed as:

$$C_{out}\left(\sum_j p_j L_i^{e_j} + \lambda\sqrt{\sum_j p_j (L_i^{e_j})^2 - \left(\sum_j p_j L_i^{e_j}\right)^2}\right)$$
$$= C_{out} L_{penalty,i}^{p_1,...p_n,\lambda}$$

Thus, the penalty function is directly proportional to the output capacitance and is only dependent on the state probabilities $p_1,..., p_n$. We use as our cost metric for leakage, the slope of this penalty with respect to output capacitance. A Pareto point consists of delay and the leakage penalty slope $(d, l_{penalty})$. Note that unlike the case of worst case

leakage, $l_{penalty}$ for a cell is also dependent on the input state probability of the matched pattern. Thus, this value cannot be pre-computed for library gates and must be computed during the post order traversal during the forward phase of algorithm based on the actual state probability information of the library pattern. The simulation for a subject graph can be done before the run of mapping algorithm since the signal state probabilities are independent of a specific technology binding of the subject graph.

The equations for adding and merging can be written as follows similar to (3.1):

$$d = \max(d_1,...,d_n) + T_c^G$$
$$l_{penalty} = \frac{l_{penalty,1} + ... + l_{penalty,n}}{g_i} + L_{penalty,c}^{p_1,...,p_n,\lambda}$$

The algorithm flow is similar to the case of dominant state GB-TM described in the previous Subsection except that we replace the leakage slope of the dominant leakage state by the leakage slope of the penalty function.

3.4 IMPLEMENTATION AND RESULTS

The library consists of 9 cells with different functionalities and 60 structurally different cells. For each cell we consider all possible input states and identify the dominant leakage states. For each of these dominant leakage states, we select the leaking transistors and set them all to a high Vth. Thus, for each library gate we have several mixed dual-Vth versions. Having such a configuration space of mixed dual-Vth gates allows us to reduce leakage significantly without having to pay a high delay penalty [13]. We always have two versions for each gate in which all transistors are either set to low Vth or high Vth. In all our experiments, we have set the probability of any primary output being 1 to be 0.5.

The previous work in [8] uses gain-based TM for the minimization of area under delay constraint. The technique presented is also applicable to leakage minimization

under delay constraint. The area-delay trade-off in [8] is generated by obtaining minimum delay mappings by fixing various global gain values. As we sweep across the gain in increasing order we obtain successively circuit configurations with higher delay and lower area. The mappings corresponding to the least delay for each value of global gain result in a monotonic curve of area as a function of delay. An analogous technique also provides a trade-off between leakage and delay.

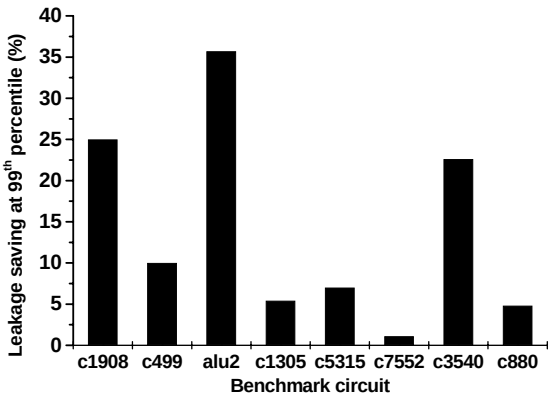

Figure 3.6 Comparison of leakage savings at 99th percentile enabled by our approach with the algorithm presented in [2]. We are able to achieve 14% savings in leakage on average.

In our experiments, the global gain *G* was varied from 1.111 to 1.120 in steps of 0.001 and the pivotal gate was chosen to be low threshold voltage NAND4 gate. These choices are arbitrary and others could have been set-up as well. In Figure 3.6, we show the percentage improvement in leakage with respect to the Pareto points generated by the mapper of [8]. The average leakage gain is 14% and the leakage gain can increase to as much as 20% for some benchmarks. Improvement is achieved at the cost of a minor (1%) penalty in delay with respect to mappings produced by [8].

In the next experiment, we contrast the dominant leakage versus the quantile GB-TM algorithms wherein both algorithms are implemented in our gain-based framework and perform simultaneous propagation of delay-leakage tradeoff Pareto points is performed in both cases. The results we obtain indicate that the dominant-state leakage minimization gives sub-optimal results compared to the penalty function based approach. This can be attributed to the ability of the latter approach to account for the impact of signal state probabilities on total circuit leakage. The comparison results are reported in Table 1. For each benchmark, we used 10 different global gain values of 1.75, 1.85, 2, 2.1, 2.25, 2.35, 2.5, 2.75, 3.25 and 3.5. Across the benchmarks, we obtain average savings of 12.8% and maximum savings of 21.49% at 99% quantile leakage. Our algorithm has extremely small run-time and provides a 15X speed-up over a bin based algorithm utilizing a discretized library. The run-time reported in the table is cumulative run-time for the 10 gain values and for a single value of penalty factor λ. We made seven choices 0.1, 0.2, 0.3, 1, 2, 3 and 0 for the penalty factor.

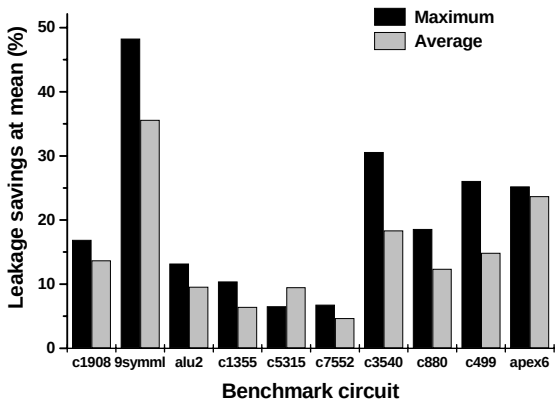

Figure 3.7 Comparison of mean leakage minimization and dominant leakage minimization. The y-axis gives savings at the mean value of leakage. We obtain on 14.5% average savings 20.5% maximum savings.

Finally, we demonstrate that it is possible to obtain significant reduction in mean leakage using the GB-TM algorithm compared with dominant state leakage minimization (Figure 3.7). The algorithm minimizes mean leakage if we choose the penalty factor of 0. On comparing the Pareto sets generated by our algorithm for mean against the Pareto point of the dominant leakage minimizing mapper, we obtain on 14.5% average savings 20.5% maximum savings in leakage across the benchmarks.

Benchmark	Size	Max leakage improvement (%)	Average leakage improvement (%)	Run Time for Gain-based Mapping (s)	Run time for Bin-based Mapping (s)
9symml	311	31	23.28	10	75
C1908	1167	17.80	12.14	60	802
C7552	4609	22.48	9.52	200	5006
C5315	3448	11.44	6.05	110	1210
C3540	1942	9.32	6.65	100	1007
C1355	706	70.12	40.60	40	362
Apex6	886	10.04	7.89	30	210
Alu2	565	12.26	6.41	20	194
C880	521	9.59	7.27	20	190
C499	711	20.98	8.92	40	434
Average	1486	21.49	12.8	63	949

Table 3.1 Comparison between the penalty function based technology mapper and the dominant state leakage based technology per. Leakage is measured at the 99th quantile.

3.5 CONCLUSION

We demonstrated that the leakage of any partial mapping has a linear dependency on the output capacitance, and implemented a gain-based mapper that uses this model. We have also showed the need for probability aware optimization.

Chapter 4. Technology Mapping for Runtime Leakage under Workload Uncertainty using Robust Dynamic Programming

4.1 MOTIVATION: INTERVAL STATE PROBABILITIES

Minimization of leakage power consumption has become an overriding concern at all levels of design. While standby leakage control has been of importance for many years, recently there has emerged a need to runtime leakage control [83][84]. These optimization techniques rely on the fact that leakage power of a combinational gates and circuits differs sometimes widely depending on the input vector. The state probabilities can be computed and can be used to minimize power. Both average and maximum runtime leakage power can be reduced by exploiting the leakage state dependency and the unequal state probabilities and leakage variances as shown in Chapter 3.

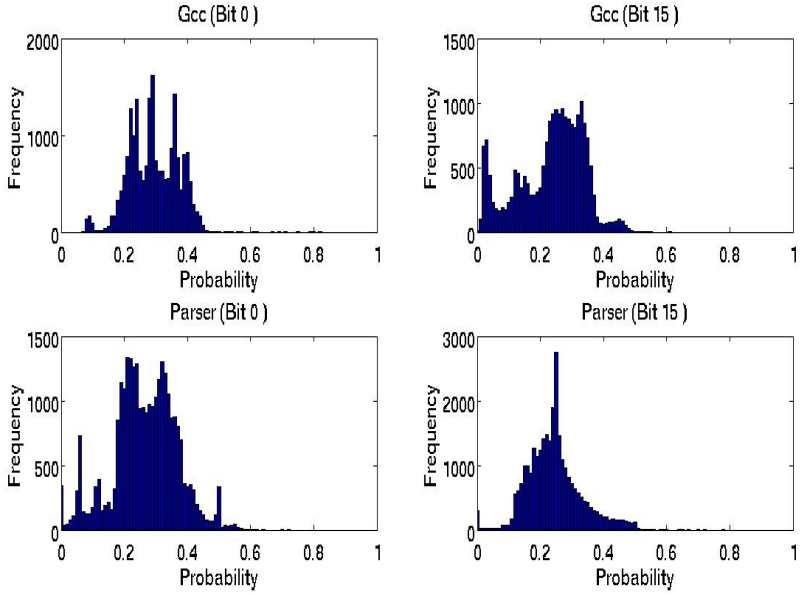

Figure 4.1 The probability histogram of two input bits of an *adder* in MIPS like architecture running `gcc`, `parser` applications. Due to wide range that the point probability spans, a single point probability is not a useful measure of node activity.

It was observed that a 62% reduction of power is possible if the assignment of transistors in a gate to higher Vth is done taking into account the probability of a gate being in a specific state: for gates with the a skewed state probability, only transistors leaking in that state will be set to high Vth, minimizing leakage with minimal delay penalty. The results of optimization are very sensitive to assumptions about input probabilities. While it is often assumed that the state probabilities are known, their precise form cannot be easily determined.

The notion of transition and/or state probabilities for a given combinational circuit is central to a large body of algorithmic work in CAD. Dynamic power cannot be accurately estimated without accurately computing the transition probabilities of a logical network [85]. The methods of computing transition probabilities can be divided into simulation based [86] and analytical probabilistic techniques [87]. The analytical techniques typically have problems in accurately handling DAG in which signals have high degree of correlation. In addition to power analysis, state and transition probabilities have been used in logic synthesis. Specifically, technology mapping algorithms for minimum dynamic power have exploited the knowledge of switching probabilities [88]. Technology mapping for minimizing mean value leakage under delay constraint has been studied in [89]. Chapter 3 also discusses a gain-based formulation for addressing this problem. The dynamic programming approach is employed and delay-leakage Pareto points are propagated and alpha-approximate pruning principle [17] is used to exponentially limit the growth of generated propagated point in terms of the sub-optimality incurred.

In this work we first subject to critical scrutiny the validity of the notion of well-defined state probabilities for combinational circuits.

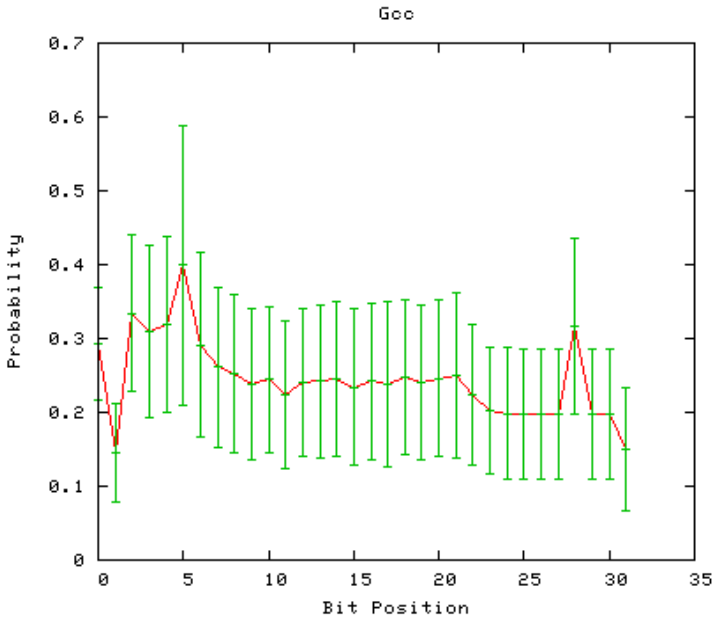

Figure 4.2 We depict the $mean \pm sd$ of the point probabilities of the 32 bits of an adder circuit in a microprocessor computed using the running average of $5 * 10^4$ samples. The number of times this average was measured is 10^4, thus spanning $5 * 10^8$ cycles. The application is `gcc`.

All the known techniques for evaluation and synthesis based on probability computation assume that the primary input probabilities are given and thus are known precisely. For microprocessors, however, the input vector probabilities are determined by the type of the workload, or software that is being executed by the microprocessor. In general one may expect that on a circuit running multiple applications, or even with a single application during different phases of the program, various different signal state probabilities will be exhibited. In this work, we show experimental evidence for this claim.

Table 4.1 Mean and standard deviation of the point probability of two input bits of an adder is shown.

Bench	Bit 0		Bit 15	
	Mean	SD	Mean	SD
bzip2	0.44	0.18	0.23	0.21
gcc	0.29	0.08	0.23	0.11
gzip	0.41	0.11	0.15	0.07
mcf	0.24	0.16	0.36	0.14
parser	0.27	0.10	0.24	0.09
twolf	0.51	0.03	0.24	0.02

We argue that the more useful way of representing state probabilities is via the probability ranges or intervals. The actual value of the input probability can take potentially any value within these intervals. We argue that under such interval description of state probability, one can utilize a *proxy leakage metric* which we define to be the sum of the worst-case mean leakages of all the gates under the feasible probability scenario. We propose TM algorithm under the delay constraint such that proxy metric is minimized across all the feasible realization of the primary input state probabilities. This metric helps to reduce the sensitivity of the leakage optimized mapping to a specific point probability for the primary input signals. We solve the TM problem utilizing robust dynamic programming [90]. The robust technology mapping tool that we develop will describe input probabilities as belonging to families of distributions.

The computational challenge of dynamic programming under uncertainty is addressed by adopting the robust (set-theoretical) modeling paradigm. Robust DP can be cast as a discrete Markov decision process with uncertain transition probability matrices. The algorithm involves an inner loop to estimate and propagate the robust cost. Under several reasonable models of uncertainty (ellipsoidal, matrix-interval), the inner loop of

DP can be solved efficiently, so that robustness is achieved with little cost. A key issue for a practical algorithm is its computational performance. Many problems of decision-making under uncertainty are computationally hard, including stochastic programming and stochastic dynamic programming [92]. A continuous stochastic dynamic programming problem can always be turned into an approximate Markov chain process, known as discrete Markov Decision Process (MDP) which is, however, amenable to efficient solution. By formulating the dual DP problem, recent work demonstrated the general possibility of computationally tractable robust dynamic programming for several classes of convex uncertainty sets [91].

We provide experimental data to show that the point probability cannot capture the pattern of inputs observed in a typical microprocessor. We estimated the state probability of different pins of an adder circuit in the microprocessor by taking running average of consecutive $5 * 10^4$ iterations. The number of times this average was measured is 10^4, thus spanning $5 * 10^8$ cycles . We used a functional microprocessor simulator (sim-safe) from the Simplescalar [95] toolset to gather the data. SimpleScalar toolset is a system software infrastructure widely used for program analysis, microprocessor design and verification. The simulator executes unmodified binaries compiled using gcc for a MIPS like Instruction Set Architecture. We modified the simulator to gather the input values of integer add and subtract instructions. We used our modified simulator to gather the probability data for 6 SPEC2000 [96] Integer benchmark programs. These are a set of industry standard programs released by Standard Performance Evaluation Corporation (SPEC) and are universally used in the design and evaluation of new microprocessor architectures. These programs have characteristics that are representative of real world applications typically run on a computer. The applications chosen were: bzip2 (data compression), gcc (c-compiler, gzip (gnu data compression), mcf (combinatorial

optimization), parser (word processing) and twolf (placement tool). We used MinneSPEC [97], a set of reduced input data files for SPEC2000 in our simulations. In Table 4.1, the mean and standard deviation of the point probabilities for two different input bits of an adder are shown. It can be seen that the mean value of the state probability varies widely across different applications. Even for many single applications, there is a significant fluctuation in the estimated state probability as indicated by the standard deviation (SD). Figure 4.1 and Figure 4.2 also corroborate this conclusion.

The impact of interval probabilities on the leakage is significant. To validate this we produced several hundred of mapped circuits for the benchmarks 9symml.blif, C880.blif and C3540.blif. We test two cases. In the first case we applied a point probability of 0.5 at each of the primary input and second case we assume an interval probability of [0.3, 0.7] for each primary input. The mean leakage and proxy leakage metric are compared for the first and second case respectively. The differences are found to be 30%, 52% and 34% respectively.

4.2. PROPAGATION OF INTERVAL PROBABILITIES

Any algorithm involving leakage optimization must require the knowledge of intermediate signal state probabilities in order to correctly evaluate the mean circuit leakage. Thus, an important step of the algorithm is the estimation of the uncertainty interval of the state probabilities of the intermediate circuit signals.

The exact estimation of the point probabilities of the intermediate signals for the case of known primary input point probabilities is a known NP-complete problem. Various approximate approaches have appeared in literature, such as analytical techniques [99][100] and simulation based approaches to estimate the signal's point probabilities.

Table 4.2 The distribution of errors in estimating probability intervals. A significant portion has been predicted with error of less than 0.1 and 0.2

Circuit	$error \in [0, 0.1]$	$error \in [0.1, 0.2]$	$error \in [0.2, 0.3]$
C3540	71.35/77.94	26.63/20.24	1.85/1.64
C1908	77.35/81.98	19.98/15.26	2.65/2.57
C499	68.87/74.64	25.35/20	5.49/5.35
C880	73.46/82.5	23.84/15.19	2.69/2.3
Alu2	73.35/79.92	24.33/17.93	2.3/2.13
Alu4	70.17/78.94	28.07/14.03	1.75/7.01
Apex6	63.84/75.14	35.84/23.84	0.67/1.01
C1355	78.29/69.07	14.89/19.71	6.8/11.2
C6288	43/38.35	44.48/39.4	8.56/19.89
C7552	55.01/63.99	35.67/27.53	7.66/7.24

We propose an algorithm for interval probability estimation based on propagation of affine combination of a fixed set of intervals describing the uncertainty at primary inputs. Let the subject graph which is an input to the TM algorithm be comprised of

NAND and INV gates, and has n primary inputs. Let the interval probabilities associated with the ith input be $Z_i = [a_i, b_i]$. In the first step we observe the following

$$[a_i, b_i] = \frac{a_i + b_i}{2} + \frac{b_i - a_i}{2} * [-1,1] = p_i + q_i U_i$$
$$\text{where } p_i = \frac{a_i + b_i}{2},\ q_i = \frac{b_i - a_i}{2},\ U_i = [-1,1]$$

We approximate the signal probabilities as linear combination of $U_i's$. As shown above the primary input probabilities have already been expressed as affine combinations. Now consider an inverter with input probability $b_0 + \sum_{i=1}^{n} b_i U_i$, the interval state probability for the output of the inverter has the affine form $(1 - b_0) + \sum_{i=1}^{n} (-b_i) U_i$. In case of NAND gate, suppose the input probabilities are expressed as $p = b_0 + \sum_{i=1}^{n} b_i U_i$ and $q = c_0 + \sum_{i=1}^{n} c_i U_i$, the output state probability is $1 - pq$. We give the affine approximation for pq based on which $1 - pq$ can be recovered. The approximation for pq is $b_0 c_0 + \sum_{i=1}^{n} (b_i c_0 + c_i b_0) U_i$. We essentially ignore the cross-product terms. In [98] an alternative approximation for taking products is formulated which introduces a new independent random variable in each multiplication. The interval estimated will always cover the true interval. However this method has run-time as well as memory implication for the circuits containing a large number of gates. Experimentally we observed that the proposed method of neglecting the cross-terms performs better compared to [98] . We report the results in Table 4.2. More work is needed to further improve the probability interval propagation. For our experiments (Section 4.5) we used the Monte Carlo simulation methodology in order to maintain high accuracy.

4.3 BASICS OF GAIN BASED TM FOR LEAKAGE

First, we briefly recapitulate the gain based TM proposed in Chapter 3 which we have enhanced to develop the robust mean leakage minimization TM algorithm. The gain based TM has several advantages over the bin based mapping such as faster run-time, guarantee that the delay constraint will be met, works for nearly continuously sized library.

We are given a subject graph which is a network consisting of NAND and INV gates, and a library consisting of a set of gates that are also represented in canonical form. There are n primary inputs $i_1,\ldots,i_n$. Let the probabilities of these signals being equal to 1 be $p_1,\ldots,p_n$ respectively.

We assume that these probabilities are independent of each other at the primary inputs. In what follows we develop the necessary models and algorithms for gain-based TM. The algorithm minimizes mean leakage with delay constraint.

We assume a continuous library of N gates. A logic effort model of delay is adopted [34]. In the gain based model, the delay of each gate is characterized as $d = g_i h_i + p_i$, where g_i is the electrical effort C_{out} / C_{in} and h_i is the logical effort and p_i is the parasitic delay. The logical effort theory [34] states that the delay of a path is optimized if the effort delay $g_i h_i$ is balanced along the path. One way to achieve this is to define a global gain value G for the inverter or some other fixed library cell which we refer to as pivotal gate and then determine the gain value of all the cells according to (the notion of global gain has also been employed in context of gain based min area mapping in [8])

$$g = \frac{G.h_{inv}}{h}$$

Let ith cell delay, capacitance, dominant leakage, width, gain be denoted as $t_i, C_{in,i}, leak_i, W_i, g_i$. We extract using SPICE simulation the following information for cell i

$$t_i = h_i g_i + p_i = h_i \frac{C_{out}}{C_{in}} + p_i \text{ , } C_{in,i} = r_i W_i \text{, } leak_i = q_i W_i .$$

In the above equations we define q_i / r_i to be the leakage slope coefficients. This multiplied with the input capacitance gives out the leakage and it is different for different states of the same library gate.

After we fixed the global gain G, the delay of each cell is fixed and the delay of i-th cell is given by $T_i^G = G.h_{inv} + p_i$. For output capacitance at i-th cell, C_{out} we can estimate the leakage in terms of the output capacitance as follows:

$$T_i^G = \frac{C_{out}}{C_{in}} h_i + p_i, \frac{T_i^G - p_i}{h_i} = \frac{C_{out}}{C_{in}}$$
$$C_{in} = C_{out}\left(\frac{h_i}{T_i^G - p_i}\right), W_i = C_{out}\left(\frac{h_i}{r_i(T_i^G - p_i)}\right)$$
$$leak_i = C_{out}\left(\frac{q_i h_i}{r_i(T_i^G - p_i)}\right) = L_i C_{out}$$

Note that now leakage is expressed as a linear function of output capacitance. The mapping algorithm takes the aforementioned library characteristics as input parameters. The uncertainty in the input vector state is captured by specifying the probability of each primary input being one. Thus leakage itself becomes a random variable due to the state dependency. Gain-based technology mapping minimizes the mean value of the leakage.

$$\min E(leakage)$$
$$d_{circuit} \leq d_{given}$$

In order to map a circuit such that leakage at mean value is minimized (due to the gate state uncertainty), we would like to characterize the leakage of library gates for each input vector instead of the worst case leakage. Let the ith library gate has x_i inputs. Thus

it has $y_i = 2^{x_i}$ input states. In an input state e, let the leakage can be characterized as $leak_i^e = L_i^e C_{out}$. It can be shown that the mean leakage is also a linear function of the output capacitance. We illustrate its computation assuming that the gate i is driving output capacitance C_{out}. Let the input states $e_1,...,e_n$ have probabilities $p_1,...,p_n$. Hence, the mean value can be computed as:

$$C_{out}(\sum_j p_j L_i^{e_j}) = C_{out} L_i^{p_1,...p_n}$$

In order to find the best leakage mapping under delay constraints, we propagate a set of delay and leakage slope Pareto points (D, L_{slope}). These Pareto points represent the cost of competing partial mappings. Let $(d_1,l_1),...,(d_n,l_n)$ be a set of Pareto points at the first to n-th input of a pattern p with gain g whose cost itself is (d,l). The partial mapping obtained by merging these Pareto points and adding the pattern p has delay, leakage slope (D, L_{slope}) computed as follows:

$$\begin{aligned} D &= \max(D_1,...,D_n) + d \\ L_{slope} &= \frac{L_{slope,1} + ... + L_{slope,n}}{g} + l \end{aligned} \qquad (4.1)$$

The rationale for dividing by g can be observed as follows. Let C_{out} be the capacitance p is driving, then its input capacitance is C_{out} / g, which, in turn, is the output capacitance seen by the mappings rooted at its n inputs. Hence total leakage is

$$\begin{aligned} &L_{slope,1} C_{out} / g + \cdots + L_{slope,n} C_{out} / g + l C_{out} \\ &= (L_{slope,1} + \cdots + L_{slope,n} / g + l) C_{out} = L_{slope} C_{out} \end{aligned}$$

The Pareto points are propagated during the post-order traversal using the operations defined in (4.1) until we reach the dummy sink node which has zero delay and zero leakage slope coefficient.

Based on the delay constraint the mapped circuit is recovered during a pre-order traversal of the subject graph. During the pre-order traversal we propagate backwards the

arrival time constraints starting from primary outputs and select library patterns that are binded to specific subject graph nodes so that we cover the full subject graph. We always choose the Pareto points which satisfy the arrival time constraints with lowest value of leakage slope. For example if we encounter a Pareto point M which corresponds to pattern c which has delay d. Suppose M has arrival time constraint a. The arrival time constraints at M's fanins become a-d. We call a-d as the arrival time imposed by M. At each fan-in we pick up a Pareto point with lowest leakage slope which satisfies the arrival time constraint imposed by M. In case of multiple primary outputs the quality of the mapped circuit depends on the order in which we process the primary outputs. It is usually most beneficial to do pre-order traversal in the order of the most critical (in terms of available slack) to least critical primary output [89]. In case of multiple primary outputs with path reconvergences, we may run into scenario where during the post order traversal some fan-in of the pattern associated with the current Pareto point M may be already mapped. Suppose this fan-in cone is rooted at subject graph node v. In this case we check if the currently mapped Pareto point at v meets the arrival time constraint imposed by the M. In this case we don't change the mapped pattern assigned at that fan-in. Otherwise we replace it by the best leakage slope Pareto point rooted at v which meets the new arrival time constraint. This is accomplished in two steps – (1) removal of the previous pattern p binded at v and recursive update of fanin cones of p since potentially they can be improved due to the removal of the arrival time imposed by p (2) Assignment of new Pareto point Q rooted at v which meets the required time and recursive propagation of arrival time constraint at the fanin cones of the children of pattern associated with Q.

4.4 ROBUST TECHNOLOGY MAPPING

We want to optimize the worst case expected value over all possible realizations of the input vector probabilities satisfying the input interval constraints. We first produce the interval uncertainty estimates at all the intermediate signals of the subject graph. Let the circuit G have n primary inputs and i-th primary input has associated probability interval $[a_i, b_i]$. One can construct the interval probability for all the signals based on this information. Thus for each gate in the circuit the feasible probability scenario for its various states has been characterized. Formally the problem we address is minimization of the sum of worst case mean leakage of all the gates across the feasible probability scenario.

$$\min \sum_{g \in G} \underbrace{\sup_{p_i \in [a_i, b_i]}} \{\mathrm{E}[\mathrm{P}_{\mathrm{leak,g}}]\}$$
$$s.t$$
$$D_{circuit} \leq d_{given}$$

We cast this technology mapping problem as a robust dynamic programming framework [90]. We begin by giving a brief description of the robust dynamic programming formulation used in [90] and then provide the transformation from the technology mapping setting to the formulation of robust dynamic programming in Section 4.4.

A nominal (non-robust) dynamic programming problem can be formulated in terms of a finite-state, finite action and finite horizon Markov Decision Processes. In a finite horizon MDP, we have a finite set of decision horizons $T = (0,1,...,N-1,N)$. At each stage the system occupies a state $i \in \mathrm{X}$, where the state space size $|\mathrm{X}|$ is finite and equal to n. A decision maker is allowed to choose an action $a \in A$ from a finite set of allowed actions. At any decision horizon, the state makes a Markov decision according to a collection of transition matrices $\tau = \{P_t^a \mid t \in T \wedge a \in A\}$. In other words, a state i at

decision horizon t makes a transition to state j with probability $P_t^a[i][j]$ if the decision maker chooses the action a. Probabilities defined in the transition matrix are set to 0, if the corresponding transition is not allowed. There is a cost associated $c_t(i,a)$ if an action a is selected at decision horizon t when the state was i. In the final decision stage N, there is no transition made to the next stage. . If $t = N$, we do not make any transition, hence the cost is just a function of state $c_t(i)$. The cost is assumed to be always non-negative and finite. A controller policy is the assignment of actions corresponding to each state and decision horizon. Formally a controller policy $\pi = (a_0,...,a_{N-1})$ assigns action $a_t(i)$ to state i at decision horizon t. Given a controller policy π, we can define $v_t(i,\pi,\tau)$ as the mean value of total costs that is incurred if we start from the state i in the decision horizon t and follow afterward the controller policy. We call $v_t(i,\pi,\tau)$ the cumulative mean cost starting in state i, at time t. The $v's$ can be recursively computed as follows:

$$\begin{aligned} v_t(i,\pi,\tau) &:= c_t(i,a_t(i)) + P_t^a[i][0].v_{t+1}(0,\pi,\tau) + ... \\ &\quad + P_t^a[i][n-1].v_{t+1}(n-1,\pi,\tau) \quad (4.2) \\ v_N(i,\pi,\tau) &:= c_N(i) \end{aligned}$$

Let Π be the space of all controller policies. Let $'root'$ be defined as the initial state at the timing horizon 0. The finite-horizon nominal dynamic programming problem can be described as the determination of the policy that minimizes the expected total cost starting from the initial state,

$$\phi_N(\Pi,\tau) := \min_{\pi\in\Pi} v_0('root',\pi,\tau)$$

The output of the algorithm is the optimum controller policy π^* and the optimum means cost value. The problem can be recursively solved using the standard Bellman recursion [90]:

$$\phi_N(\Pi,\tau) = \min_{\pi\in\Pi} v_0('root',\pi,\tau) = v_0^*('root')$$
where
$\pi^* = (a_0^*,...,a_{N-1}^*)$ [the optimal controller policy]
$v_t^*(i)$ [optimal cumulative mean cost in horizon t at state i]
$$v_t^*(i) = \min_{a\in A}(c_t(i,a) + P_t^a[i][0].v_{t+1}^*(0) + ... + P_t^a[i][n-1].v_{t+1}^*(n-1))$$
$$a_t^*(\mathrm{i}) \in \arg\ \min_{a\in A}(c_t(i,a) + P_t^a[i][0].v_{t+1}^*(0) + ... + P_t^a[i][n-1].v_{t+1}^*(n-1))$$
$$v_N^*(i) = c_N(i)$$

The application of nominal dynamic program requires knowledge of transition probabilities. When this information is unavailable and state transition probabilities span a range of values, nominal DP may produce results that become non-optimal under small perturbations of assumed conditions. The robust dynamic programming paradigm addresses this limitation. Essentially we seek to minimize the worst case expected value that can be realized under all the transition probabilities that are feasible. There are various uncertainty models which allow one to encode the multiple collections of transition probabilities such as likelihood model, interval model, entropy models etc. [90]. We will adopt the interval model of uncertainty. Thus instead of the transition matrix P_t^a we posit interval probability constraint from each state i to other states j as follows:

$$L_t^a[i][j] \le P_t^a[i][j] \le U_t^a[i][j]$$

First we formally state the robust dynamic problem. We let T_t^a denote the set of transition matrices allowed by the above interval model of uncertainty for the transition matrix P_t^a. Let set T be the union of T_t^a.

$$\phi(\Pi,\mathrm{T}) \coloneqq \min_{\pi\in\Pi}\max_{\tau\in T} v_0('root',\pi,\tau)$$

In [90] authors present a modified Bellman recursion scheme to solve this robust dynamic program:

$$\phi(\Pi, T) = \min_{\pi \in \Pi} \max_{\tau \in T} v_0('root', \pi, \tau) = v_0^*('root')$$

where

$\pi^* = (a_0^*, ..., a_{N-1}^*)$ [the optimal controller policy]

$v_t^*(i)$ [worst case optimal value function in horizon t at state i]

$$v_t^*(i) = \min_{a \in A}(c_t(i,a) + \max_{L_t^a[i][i] \le P_t^a[i][j] \le U_t^a[i][j]}(P_t^a[i][0].v_{t+1}^*(0) + ... + P_t^a[i][n-1].v_{t+1}^*(n-1)))$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{\text{INNER LOOP}}$$

$$a_t^*(\text{i}) \in \arg\min_{a \in A}(c_t(i,a) + \max_{L_t^a[i][i] \le P_t^a[i][j] \le U_t^a[i][j]}(P_t^a[i][0].v_{t+1}^*(0) + ... + P_t^a[i][n-1].v_{t+1}^*(n-1)))$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{\text{INNER PROBLEM}}$$

In comparison to the Bellman recursion, this recursive computation of optimal cost has an extra inner loop which estimates the worst case cost for each action a taken in state i at time step t over the feasible transition matrices set. This worst case cost is the true metric to judge the controller policy. The outer problem tries to find the best choice of action based on the comparison of this worst case cost. The formal argument for the correctness has been provided in [90].

Note that we defined $P_t^a[i][j]$ to be the transition probability from state i, at the decision horizon t, to state j in presence of action a. Hence, we are restricted by the implicit constraint that $\sum_{j=0}^{n-1} P_t^a[i][j] = 1$. However, the definition of cumulative cost in (4.2) makes the problem well defined for arbitrary positive entry matrices P_t^a, and thus the same solution strategy will work. In the general scenario, we call $P_t^a[i][j]$ the multiplier associated with the transition from state i at time-step t to the state j.

4.5 CONVERTING TECHNOLOGY MAPPING TO RDP FORMALISM

First we consider the case of finding the mapping with the optimal mean leakage ignoring delay under point state probability. Later extensions to interval probabilities and delay constraint are provided.

Let S be a subject graph consisting of NAND and INVERTER base gates. In order to reformulate the TM problem in terms of MDP problem stated in Section 4.3, we need to define the following: (1) state space, (2) action space, (3) rules for generating successor state from a given state based on certain action, and (4) the cost associated with actions chosen in various states.

Corresponding to each subject graph node n we associate a state $(n,\ l,\ c)$, where $l,\ c$ are real numbers representing leakage and capacitance respectively. We refer to $n,\ l,\ c$ as the first, second, and third component of state s. The decision horizons are implicitly associated with the first component and we do not make explicit reference to them henceforward. If logic depth of a circuit is N and subject graph gate n is at logic depth k, then the associated timing horizon is $N-k$. Dummy primary input nodes are the final decision horizons corresponding to N.

Initial state is $('root',\ 0,\ 1)$ corresponding to the dummy sink. The action set consists of the all library patterns. Next we describe the rules for the state evolution from $(n,\ l,\ c)$ where the action is a_p corresponding to a pattern p. Let p have i inputs and $k=2^i$ input combinations with different leakage values. Let the probability of theses input combinations be $p_1,...,p_k$ and the respective leakage slope coefficients are $l_1,...,l_k$. Further assume that the gain of pattern p is g. Suppose $n_1,\cdots,n_i$ are the subject graph nodes rooted at the i inputs of p. The successor states are $(n_x, l_j\ /\ i, cg)$ where x ranges in integers $[1,\ i]$ and j ranges over integers $[1,\ k]$. The associated multiplier is p_j (see Figure 4.3). The assigned cost of the state is set to be $l\ /\ c$.

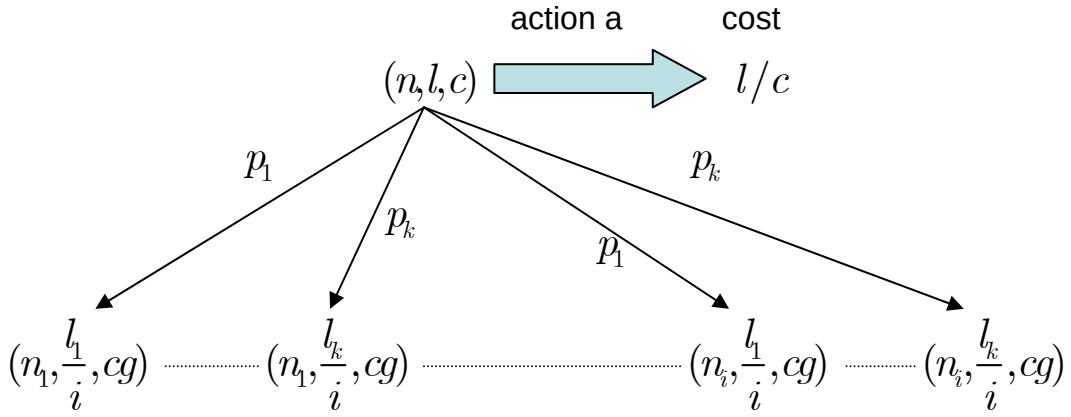

Figure 4.3 The state evolution when action a has been taken.

Next we describe the reason for equivalence of this formulation to TM formulation. Note that as we evolve the states starting from the initial states, the actions become associated with the subject graph nodes. Hence partial coverings get generated starting from the root node. As we come to a new node n, with associated state $(n,\ l,\ c)$, during the state evolution, we claim that the output capacitance that it will be seeing is C_{out}/c, where the output capacitance at the primary output is C_{out} (for simplicity we assume that it is same for all the primary outputs). This can be proved inductively. For the initial state $(root,\ 0,\ 1)$, the output capacitance clearly is $C_{out}/1 = C_{out}$. For induction hypothesis, suppose node n with state $(n,\ l,\ c)$ is driving output capacitance of C_{out}/c, consider a successor state $(n_x, l_j/i, cg)$ obtained by application of action a_p associated with pattern p with gain g. Since p is seeing output capacitance of C_{out}/c, clearly its own input capacitance is C_{out}/cg, which we needed to establish.

We also can show that the expected value of the leakage of pattern p is captured by the total costs incurred due to the successor states of $(n,\ l,\ c)$. Consider a successor state $(n_x, l_j/i, cg)$ where $j \in [1,k],\ x \in [1,i]$. The cost incurred due to $(n_x, l_j/i, cg)$ will be l_j/icg. The associated multiplier is p_j, hence the cost contribution is = $p_j l_j/icg$ (from

Equation (5.1)). If we sum this quantity over all $j \in [1,k],\ x \in [1,i]$, we get $\sum_{j=1}^{k} p_j l_j / cg$, which when multiplied with C_{out} is the mean value of leakage associated with the pattern p, since the input capacitance of p is C_{out} / cg. Thus the entire cumulative mean cost of MDP state evolution starting from the initial state will be the correct mean value of circuit leakage.

We now establish the following two properties of the MDP states that share the same first component n.

Lemma 4.1: Let $s \equiv (n,\ l,\ c)$ is a state associated with a subject graph node n. For fixed $n,\ l$ and fixed controller policy the cumulative mean cost of the entire fan-in cone of s is inversely proportional to c.

Lemma 4.2: The optimal leakage mapping of the state s is independent of $l,\ c$.

Based on Lemma 4.1 and 4.2, we can deduce that even though from a given state there are multiple successor states sharing the same first component, still they have the same optimal covering, thus avoiding any ambiguity.

Propagating both delay and leakage costs: Now we provide an extension to handle propagation of delay costs also. In this scenario, we need to propagate secondary delay costs in addition to the original leakage cost.

The new cost is not a single number; rather it is a (delay, leakage) pair. Thus, if in a state $(n,\ l,\ c)$ action a_p corresponding to a pattern p is taken, then the corresponding cost is $(d_p, l / c)$, where d_p the delay of pattern is p.

To construct the solution of delay-constrained leakage optimization problem, a Pareto set is propagated instead of merely propagating just a single cumulative mean leakage cost. The Pareto points are obtained at s on action a_p associated with a pattern p by merging the Pareto points stored at its successor states. The delay cost d is obtained

by taking the max of the delay component of the Pareto points associated with the successor states of s and adding it to the delay cost of p. A Pareto point can be dropped if it is inferior both in delay and leakage. Each Pareto point $pp = (d,\ k)$ generated at a state $s \equiv (n,\ l,\ c)$ has an associated partial covering C_{pp} at the fan-in cone of n with the delay and leakage costs being d and kC_{out}, where C_{out} is the capacitance facing the primary output. We have the following properties analogous to Lemma 4.1 and 4.2.

Lemma 4.3: The Pareto points generated at $s \equiv (n,\ l,\ c)$ have the general form $(d_i, (l + l_i)/c)$ where d_i, l_i are completely independent of l and c.

Lemma 4.4: The partial mappings associated with the Pareto points gathered at the state $s \equiv (n,\ l,\ c)$ are independent of l and c.

Lemma 4.5: Suppose during the Pareto point generation for the state s, when the merge operation is performed and two successor states share the first component (hence have the same mappings associated with the Pareto set). If we use Pareto points corresponding to different partial covering for these successor states in the merge operation. In such a scenario there is sub-optimality incurred during the merge operation.

Based on the above lemma we conclude that Pareto points set associated with a state $(n,\ l,\ c)$ needs to be constructed only for one choice of $l,\ c$ and for other choices it can be recovered by scaling (Lemma 4.3) and the associated partial mappings do not change (Lemma 4.4). During the merge operation, where there are multiple successors sharing the same first component, the ambiguity is avoided by merging the Pareto points that correspond to the same mapping (Lemma 4.5).

The mapping is recovered using backward traversal from the root node. Essentially we look at all the Pareto points generated at the root node. Find the one which has the smallest leakage cost under the user-specified delay constraint. Afterwards, we look at the action that was used to generate that Pareto point. The action is essentially is a

pattern. We look at the fan-ins of this pattern and recursively continue backward traversal propagating the required arrival time and selecting best leakage coverings using the algorithm similar to the case of Gain based mapping (Section 4.2).

Robust Version: In this scenario, the multipliers associated with the state transitions are not point values but are modeled as intervals. From the previous discussion we know that the multiplier associated with the state transition from the state $(n,\ l,\ c)$ to $(n_x, l_j / i, cg)$ is p_j. Also p_j is the probability of certain input combinations $\varepsilon_{1j},...,\varepsilon_{ij}$ at the inputs of the pattern p. Let the state probabilities at the inputs of pattern p be $q_1,...,q_i$ with $q_m \in [a_m, b_m]$. Let $\pi(\varepsilon_{mj})$ be q_m if ε_{mj} is 1 and otherwise $1 - q_m$. In this case we have $p_j = \prod_{1 \le m \le i} \pi(\varepsilon_{mj})$. It can be shown that the inner loop problem becomes

$$\begin{aligned} &\sup \quad \sum_{j=1}^{k} l_j p_j / cg \\ &s.t. \quad p_j = \prod_{1 \le m \le i} \pi(\varepsilon_{mj}) \\ &where\ j \equiv (\varepsilon_{1j}....\varepsilon_{ij})_2 \\ &\qquad q_m \in [a_m, b_m] \end{aligned}$$

Lemma 4.6: An optimal solution to the above problem must have for all m, q_m equal to either a_m or b_m.

Corollary: The solution to the above problem can be found by exhaustively enumerating all the 2^i possibilities for q_m taking either a_m or b_m. Since i is the number of library gate pins which are usually very small number, this method can be used to find the worst case value of the leakage in a very small amount of time.

4.6 EXPERIMENTS

The TM library consists of 9 logically different cells and overall 50 structurally different cells. For each logically different cell we consider all possible input states and identify dominant leakage states. For each dominant leakage states, we select the leaking

transistors and make a group of them. We only raised Vt of leaking transistors. Having such mixed-Vt gates allow us to reduce leakage significantly without having to pay-off a high delay penalty caused by the introduction of all high-Vt gates [84]. Using the same library, we run a robust version of algorithm assuming an interval probability of [0.3, 0.7] and other experiment with a point probability of 0.5. The global gain was chosen from 0.8 to 1.7 in increments of 0.1. The upper limit on the number of Pareto points propagated was kept to be 50. The experiments have been run on the circuits in the publicly available ISCAS'85 and MCNC benchmark suites. The library was characterized for a 70 nm process using Berkeley Predictive Technology Model. The experiments were carried out on a Sun SPARC 1.2GHz, 32 bit machine.

The comparison is made based on the interval probabilities evaluated using the Monte Carlo simulation. Essentially we accumulate the Pareto point generated by both the robust and point probability mapping algorithms and find their union. Then we find the Pareto points in this union and for each delay/leakage trade-off point $(d,\ l)$ remaining in the robust TM output corresponding to a mapping M_1. We also get the best leakage mapping M_2 constrained by delay d in the point probability based TM. The comparison is done by estimating the leakage improvements for the point probabilities chosen from 0.1 to 0.9 for all the primary inputs. The numbers of Pareto points generated were of the order of several hundreds. In Figure 4.4 to Figure 4.6, we reported for three benchmarks the average leakage improvement, max improvement achieved by the robust mapper for various test point probabilities at the primary inputs. The generic trend is - as we deviate more from the central point probability the improvement becomes more reaching upto 45%, 32% and 17% respectively in the benchmarks *tree*, *C880*, *C1355*. The summary for all the benchmarks is depicted in Figure 4.7. We can obtain average improvement

upto10-16% depending on ± 0.3 shift from this point probability when the shift is ± 0.1 we can obtain improvement upto 6%.

4.7 CONCLUSION

We have presented evidence that the signal state probabilities are not point values and best modeled as an interval. Moreover there is a significant difference in the mean leakage assuming approximate point probability and robust value of leakage which necessitated the development of efficient robust leakage minimization under delay constraint. We have proposed such an algorithm and obtained improvement of $9 \pm 4\%$ in ISCAS'85 benchmarks.

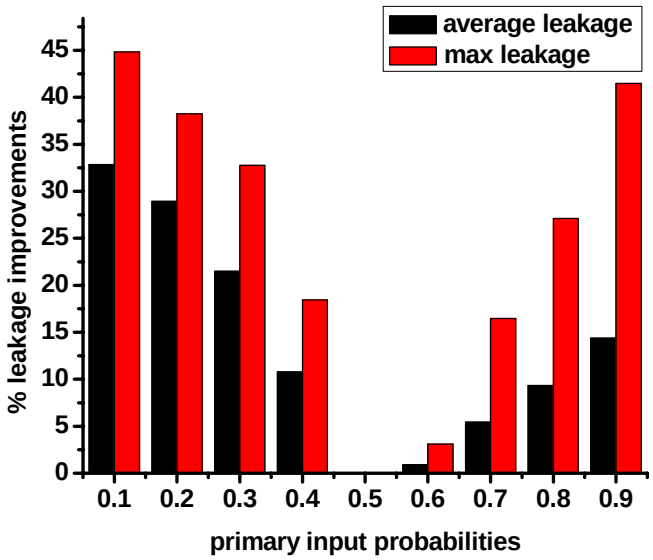

Figure 4.4 The average and max leakage improvements for different test probabilities ranging from 0.1 to 0.9. The benchmark is a tree circuit.

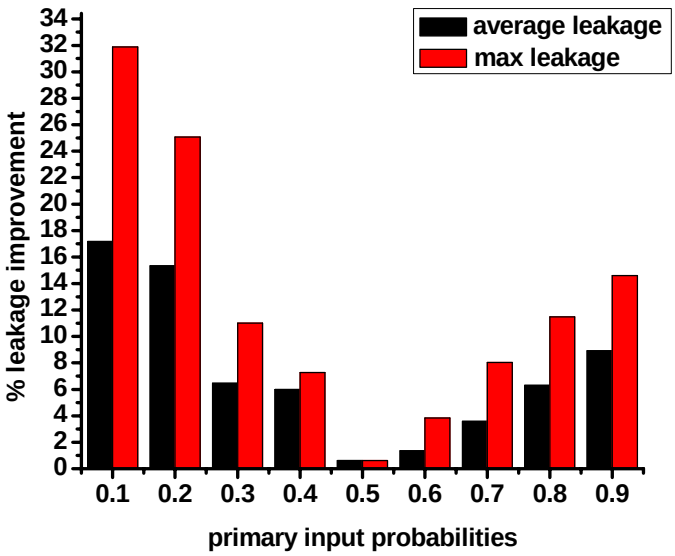

Figure 4.5 The average and max leakage improvements for different test probabilities for benchmark C880.

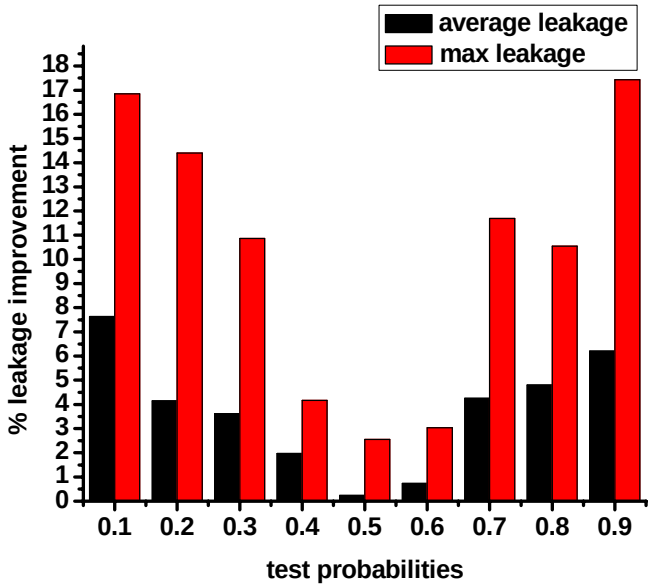

Figure 4.6 The average and max leakage improvements for different test probabilities ranging from 0.1 to 0.9 for C1355.

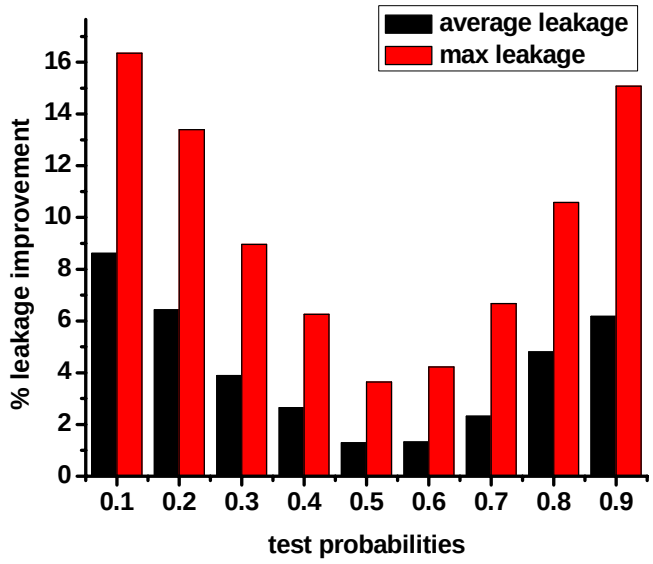

Figure 4.7 The aggregate result for all the benchmark. The primary input probabilities ranging from 0.1 to 0.9.

Chapter 5. Generation of Efficient Codes for Realizing Boolean Functions in Nanotechnologies

The end of CMOS scaling will require finding alternatives to CMOS electronics. The currently explored technologies include carbon nanotube (CNT), molecular, and quantum devices [61] It is premature to conclude which particular device technology will ultimately prove to be most promising, although a technology based on FETs with carbon nanotubes as channels appears to be the most likely candidate for near-term acceptance [62][63].

The extreme *defect densities* and high *parametric variability* in emerging device technologies pose a fundamental new challenge which requires researching new cost-effective strategies for ensuring reliable computation [61]. Reliable computation in integrated digital circuits has been a subject of extensive research. A common paradigm for fault tolerant system design is the use of *redundancy*, such as triple modular redundancy (TMR), which uses three copies of a computational unit and arbiters employing majority voting to produce the correct value, or reconfiguration techniques [64]. Redundancy combats both permanent and transient faults, but reconfiguration helps deal only with permanent faults.

Most of the existing work is not directly relevant to finding a solution for the challenges of the future device technologies**:** (1) with the low defect densities, it was reasonable to assume during the redundancy insertion that the probability of the arbiter failing was negligible. This assumption becomes unreasonable in the case of nanotechnology, endangering the entire traditional fault tolerance edifice. (2) The traditional reconfiguration approach also fails because, with device defect density being of the order of 1-5%, the probability that a module which contains more than a few gates will operate correctly becomes so low that orders of magnitude more spares than actual

working components are needed. This is in contrast to the traditional silicon technologies, in which low defect density requires relatively few spares and allows performing reconfiguration at the module level [64].

At higher defect densities, defect tolerance must be addressed at a lower level of design, specifically, at the level of logic gates. The need to make individual gates reliable changes the traditional approach to fault tolerance in a fundamental manner. The reason is that now the complexity of the test and reconfiguration circuitry, as well as the arbitration circuitry becomes comparable to that of logic itself. Consequently, we believe that the right design style is to make individual modules more reliable by using redundancy internal to each module, and to choose an optimal combination of inter-module reconfigurability and intra-module redundancy.

Just as communication in the presence of noise can be made reliable by adding redundant code bits, logic circuits can be made reliable out of noisy gates by the use of redundant gates. The possibility of building reliable circuits from noisy gates has been studied theoretically, most famously, by von Neumann [66]. He introduced a massively redundant strategy, now known as von Neumann or NAND multiplexing. He demonstrated that, at the cost of tremendous overhead, for any Boolean function a reliable circuit can be synthesized even if individual devices fail with a fairly high probability (his bound was 11.87%). His construction inserted an exponential amount of redundant logic for all functions. Later, Pippenger refined von Neumann's results, showing that for a functional selected at random, the coding overhead was a constant factor, with high probability [67]; however, Pippenger's construction is exponential for the functions encountered in practice. In summary, these works have shown that it is possible to construct reliable circuits from individually unreliable gates, but the cost is prohibitively high, resulting in the area overhead of 10^3-10^4 of the original circuit [68].

Because of that, gate-level redundancy insertion for constructing reliable logic circuits has not been previously exploited in VLSI design and remained a theoretical possibility.

In this work we propose a design methodology for building reliable computational elements using devices manufactured in nano-technologies. The fundamental strategy of fault-tolerance is not reconfiguration, but low-level protection of individual Boolean functions through the use of efficient coding. The Boolean functions are represented in terms of ROMs or truth tables, and a novel version of the Hamming code is used to protect the functions. The proposed coding strategy exploits for the first time the structure of Boolean logic networks to produce better codes

The proposed computational architecture is based on a heterogeneous CMOS - carbon nanotube fabric. It seems certain that the early practical uses of CNT-based electronics will be built on top of the heterogeneous CMOS-CNT processes [69]. The enormous economic investments into the CMOS technology and design infrastructure provide a strong impetus to leverage the existing flows as much as possible. Interfacing CMOS and CNT technology is technically feasible; the economic viability of integration is seen by the recent commercialization of non-volatile memory circuits based on CNT integrated with CMOS. Our approach is predicated on optimally combining reliable, albeit low performance, CMOS components with high performance, albeit unreliable, CNT devices.

The remainder of the chapter is organized as follows. In Section 5.3, we describe the architecture of the heterogeneous circuit fabric in which the proposed coded Boolean functions can be decoded for achieving error correction. In Section 5.4 we describe the traditional Hamming code being applied for error correction of Boolean functions.

Section 5.5 describes the use of don't cares for reduction of redundancy required for coded Boolean functions. Section 5.6 gives a formal description of code construction and minimization of overhead which is solved by converting the problem to the Boolean satisfiability problem. Experiments for redundancy reduction and yield enhancement due to error correction are presented in Section 5.7, and in section 5.8 we summarize the key contributions.

5.1 HETEROGENEOUS FABRIC AND ARCHITECTURE

In building a reliable circuit from unreliable gates, we face a fundamental limitation that the overall reliability is limited by that of the gate driving the output (typically an arbiter). Enhancing the reliability of arbiters would greatly reduce the complexity of building reliable circuits. The heterogeneous fabric resolves this: CNT devices can be used as the base "noisy" logic which naturally exploits its advantages (low-cost high-density, low power); and CMOS gates can be used for recovery of correct response, i.e., for decoding, by utilizing a much higher reliability of silicon devices.

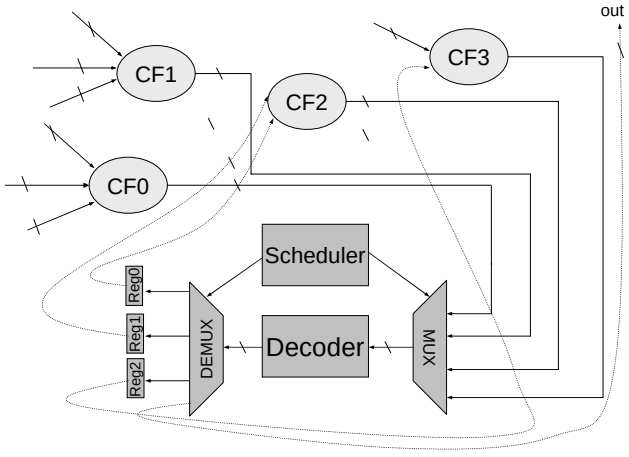

Figure 5.1 Microsequenced logic.

One simple way of mixing CMOS and nanodevices is to build *c* independent copies of circuits for each block, and use a CMOS majority voting element. This would require at least triplication of the nano-based computing blocks. We propose here a different approach that fundamentally relies on sophisticated coding.

The optimal size of nano-based ROMs and CMOS-based decoding logic is dependent on a number of technology and circuit characteristics of both fabrics. Despite the lack of clarity about CNT-based electronics, some first order estimates based on experimental results can already be made. Here we compare the key CNT properties to that of a CMOS technology at the end of the roadmap, 22nm node. The projected CMOS transistor density is from $1.2 \cdot 10^9$/cm2 for logic to $24 \cdot 10^9$/cm2 for DRAM. This is compared to substantially higher density of 10^{12}/cm2 that has been reported for a CNT-based memory array [61]. The switching time of a NMOS device in a 22nm CMOS is projected to be 0.15ps, which, surprisingly, appears to be comparable to the CNT switching time of 0.5-1ps. A significant difference can be seen in the switching energy of an NMOS device (4aJ/device) and the estimated CNT energy per switching ranging from 0.5aJ/device to 0.02fJ/device. From the above analysis, it appears that the major advantage of CNT-based circuitry over the CMOS-based circuitry is in the transistor density and switching energy, with a difference of at least a factor of 40 for density and up to 10x for switching energy. Thus, the optimal granularity of inserting CMOS-based arbiters will be limited not by the timing overhead but by the density and power constraints.

We propose to partition the logic into blocks, which individually can be implemented by a lookup-table (LUT), e.g., a ROM. The basic logic blocks will be "protected" using a coding scheme described in the next sections. By using coding at a

very low level of granularity, our technique enables a much higher probability of instantiation of a block in a defective fabric. We specifically address the challenge of protecting the content of memory bits, which are taken throughout to be the basic units of data storage in a ROM, such as in a NAND or NOR ROM architectures. The row address decoders can be protected through the use of a simple coding technique [71]. The decoder is implemented in CMOS and thus imposes area overhead compared to dense nano-LUTs. The overhead can be reduced by sharing the decoders using the time multiplexing strategy. We envision that the combinational circuit is evaluated in a sequence of microcycles, e.g., Chapter 6 of [78]. This is illustrated in Figure 1. The starting combinational circuit has four multi-input multi-ouput logic blocks. We apply our coding to each block, to produce the functions CF0-CF3. The output function is now computed by first decoding the output of CF0 and storing the result in a register. We then compute the output of CF1, and store it in a register. The inputs to CF2 are now decoded, so we take its output and decode it, storing the result in the third register. Now the inputs to CF3 are decoded, and the decoder output is the final output.

5.2 CODING OF LOGIC FUNCTIONS

Coding theory has been originally developed for communication. Codes are used to prevent corruption of signals during transmission over a channel. A key distinction between a channel and a logic circuit is that the former simply transmits bits, whereas the latter transforms their values. The transformative nature of logic operations makes the extension of coding for circuits very difficult. It has been proven that for 2-input Boolean functions other than XOR/XNOR, the best codes are repetition codes; however, this is not generally the case for more complex networks [74]. Repetition codes require duplication for error-detection, and triplication for error correction. From the view point

that is more familiar to circuit designers, repetition coding is equivalent to triple modular redundancy (TMR).

In this work, we advocate a novel approach to coding for logic. It is based on representing Boolean computation in terms of ROM-based logic. ROMs are universal computing devices that can implement any logical function. The motivation for adopting this representation is the existence of codes for spatial channels, i.e., memories, which we can exploit. Coding for memories has received significant attention in the information and coding theory communities. The study of the fundamental limits on the number of bits that can be stored in a memory with defects was pioneered by [72], with the capacity of classes of these systems determined in [73]. However, much of the prior work in this field has usually been more analytic in nature without concrete designs of codes and decoding algorithms to go with this analysis.

The design of the decoder depends on the code selected. In general, the decoder proceeds by iteratively making small changes to the word being decoded, stopping when it is a legal codeword [70]. Because the cost of decoding is important for making the proposed flow practical, we will be constraining ourselves to linear codes for which decoders are easier to build. Linear error correction code design is the art of determining appropriate subspaces of n-dimensional vector spaces with special properties. The Hamming code and its shortened versions have many special properties making them particularly suitable for nano-circuits. First, they are *perfect codes* [70] in that they can always correct exactly one error in the input bit-words. They also possess a highly intuitive structure and an elegant and simple decoding algorithm.

We propose a coding scheme for general Boolean functions, i.e., functions with n-inputs and m-outputs. Multi-output functions can be identified in multi-level logic networks [79]. We propose a coding scheme for general Boolean functions, i.e., functions

with n-inputs and m-outputs. This scheme is based on the Hamming code – a minimum number of redundant outputs are added as parity bits for coding. As is the case in coding for communication, the crucial consideration is the overhead of coding – the number of bits that need to be added. The Hamming code is optimal; a typical Hamming code is $(2^m - 1, 2^m - m - 1)$, in other words, for $2^m - m - 1$ data bits, m parity bits need to be added for full protection. A single decoder will produce a correct output for any input bitword under one bit error scenario.

Consider a Boolean function with 2 inputs (x_0, x_1) and 3 outputs $(y_0,\ y_1,\ y_2)$. The outputs are defined as $y_0 = x_0 \vee x_1$, $y_1 = x_0 \vee x_1$ and $y_2 = x_0 + \bar{x}_1$. The truth table is shown below

x_0 x_1	y_0 y_1 y_2
00	001
01	110
10	111
11	111

The standard Hamming decoder can be built for the above data by adding three redundant columns as shown below:

$$\underbrace{\begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}}_{\text{Original LUT rows + 3 redundant columns}} \times \underbrace{\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}}_{\text{Decoder\_Matrix}^{\mathrm{T}}} = [0]_{4\times 7}$$

The last three columns essentially represent the parity bits of $\{y_0, y_1\}$, $\{y_0, y_2\}$ and $\{y_1, y_2\}$ output bits respectively. The absence of an error can be detected by multiplying the output row with the transpose of decoder matrix and testing if the product is a 0-row vector. Otherwise, in a case of single bit error, one can compute the product of erroneous row with the transpose of the decoder matrix to get a row vector known as *syndrome.* The Hamming codes are special precisely because of the existence of simple syndromes: if there is an error in bit location i, then the syndrome will be identical to the *i*-th row. The cost of coding in this case is 3 extra columns added to an LUT. It is important to reduce this cost.

An important contribution that this work makes is the extension of earlier Hamming code constructions. In the next section, we exploit the fact that Boolean functions, and thus LUTs, may contain don't cares, to achieve reduction in the number of redundant columns required.

5.3 USING DON'T CARES FOR COMPACT CODING

All earlier applications of coding in fault-tolerant memory design have treated the pattern to be protected as simply a set of 1s and 0s. Here we show how the structure of Boolean functions can be used to reduce the coding overhead.

A Boolean function is defined by its ON-set, OFF-set, and its DC-set [76]. DC-set (don't-care set) is a set of inputs on which the output can be *either* 0 or 1. In control logic, many practically used functions are defined with a fairly compact ON-set, giving rise to a large DC-set. There are multiple ways to represent don't-care sets. In developing a technique for coding logic, we will rely on the LUT-based representation of Boolean functions. LUTs are essentially truth tables, and the DC-sets are captured here explicitly. For multi-level circuits, local don't care sets can be computed using standard existing techniques [76].

The key observation that we make is that it is possible to assign values to the members of the don't care set, so as to encode the logic function more compactly. Suppose we have some don’t cares in the function as shown below.

x_0 x_1	y_0 y_1 y_2
00	00X
01	11X
10	111
11	XX1

Our task is to find optimal assignments to the don’t cares so as to minimize the number of redundant columns. In this case there are 4 don’t cares remaining, which may be assigned so that we need 0 redundant columns. If we assign the first don’t care to 0 and the last three to 1, then we can construct a decoding matrix without adding any redundant columns as shown below:

$$\underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}}_{\text{Original LUT rows}} \times \underbrace{\begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}}_{\text{Decoder\_Matrix}^{\text{T}}} = [0]_{4\times 2}$$

For this example, we can correct single bit errors without any additional outputs because the only two legal outputs are 000 and 111, which have a Hamming distance of 3. The ability to reduce the cost of coding in this way is the basic intuition behind our work.

In order to exploit the existence of don't care for more compact coding, an efficient algorithm for constructing such codes needs to be developed. It follows from a reduction from graph coloring that the optimization problem for forming such codes is NP-complete. Below we describe an algorithm that casts the problem as CNF-SAT, and solves it efficiently using a SAT solver, e.g., MiniSat [77].

5.4 EFFICIENT CODING USING DON'T CARE CONDITIONS

We use a single error correcting Hamming code on each row of the LUT. Experimental results support the sufficiency of single bit error correction for realistic defect densities. A traditional Hamming code for correcting n-bit original data requires s extra *parity bits*, where s the least integer is such that $2^s \geq n + s$. The default number of extra bits required for Hamming code construction is denoted as $\triangle(n)$.One contribution of the work is *reduction* of the number of default redundant columns that one need in the traditional Hamming code by exploiting the actual data present in LUT and presence of don't cares.

Let $D_{l\times p}$ be a known Boolean matrix consisting of entries in certain LUT with l rows and p outputs. The Hamming code is described by a *decoder matrix* $H_{\lceil \log_2 p \rceil \times p}$. The decoding procedure is simple and based on the given decoder matrix. Suppose the output data which correspond to the *i*-th row of $D_{l\times p}$ has a single bit error. Let this row vector be denotes as $D_{(i)}$ and location of error bit be *j*-th starting from the left. Let the *j*-th column of H be denoted as $H_{[j]}$ then the following identity can be established which computes a row vector known as *syndrome*

$$D_{(i)}.H^T = H^T_{[j]} \quad (5.1)$$

If the syndrome vector contains 0s in all the entries, there is no error. If there is a single bit error in location j of the data then, the syndrome will be equal to the j_{th} column

of H which has a one to one mapping to the bit at position j. This allows us to uniquely identify the error and then fix it by inverting erroneous bit.

The code construction is essentially the construction of the decoder matrix *H*. In general, the code construction requires adding additional parity bits to data. When D contains exhaustively all possible Boolean vector of size p as its rows, we always need to add $\triangle(p)$ number of parity bits. However, for some $D_{l\times p}$, such as in the example previously studied, it is possible that a decoding matrix exists even without adding extra bits. We want to exploit this for constructing compact codes. Whether or not $D_{l\times p}$ requires extra parity columns can be determined by checking the following condition.

Define t to be the smallest number such that $2^t > p$. There exists a matrix $H_{t\times p}$ such that no two columns are identical and no single column of H be identically zero. Furthermore, the following condition holds $DH^T = [0]_{l\times t}$. If these conditions hold, we can design a decoder, without adding any redundant columns, which will correct a single bit error in each row of the data.

If the above condition is not met, some minimum extra columns need to be added to D, until the condition $DH^T = [0]_{l\times t}$ is satisfied for some H. Since we want to add the smallest number of extra bits, we can test the condition after adding each new column. The existence of don't cares allows greater flexibility in finding a solution to the above existence check. By proper assignment of don't cares, we increase the likelihood that fewer redundant column will be needed.

The algorithm for identification of the decoding matrix is mathematically formulated as a *Boolean satisfiability problem* which is efficiently solved using a SAT solver [77]. This formulation also permits seamless handling of don't cares in the matrix

We convert $DH^T = [0]_{l\times t}$ to the following equivalent form

$$\sum_{j=1}^{p} d_{ij}h_{kj} \equiv 0 \qquad \forall i \in [1,l] \text{ and } k \in [1,t] \quad (5.2)$$

This problem can be easily converted to a Boolean problem as follows: ($\otimes$, $\wedge$ denotes XOR and AND of two Boolean variables)

$$\bigotimes_{j=1}^{p}(d_{ij} \wedge h_{kj}) \equiv 0 \qquad (5.3)$$

Now using Tstein-transformation [75] we can convert (5.3) into a CNF form efficiently by introducing a small overhead of extra variables. The equivalent CNF can be checked for satisfiability using a standard SAT solver [75]. The further constrains on the matrix H are that no column should be identically 0 and no two columns should be identical can be easily turned into Boolean constraints as follows

$$\begin{aligned} &\bigvee_{i=1}^{t} h_{ij} \equiv 1 \quad \forall j \in [1,p] \\ &\bigvee_{i=1}^{t}(h_{ij_1}h_{ij_2}) \equiv 1 \quad \forall j_1, j_2 \in [1,p], j_1 \neq j_2 \end{aligned} \qquad (5.4)$$

These constraints can again be converted into CNF format, using the Tstein-transformation and given to the SAT solver along with (5.3). If the SAT solver detects that the set of Boolean conditions are satisfiable for some assignment to the entries of decoder matrix H, don't cares and redundant column entries of D then a feasible decoding matrix exists and we can find the entries of H from SAT solution.

The number of variables and clauses required for SAT problem is dependent on the dimension of the data matrix D, and it can be summarized as: variables including the extra variable due to Tstein transformation $(4p-2)lt + p + {}^{p}C_2(4t+1)$ clauses required: $tp + (p-1)lt + {}^{p}C_2 t$. Thus both the number of clauses and the number of rows are linear in l (the number of rows in D). Since $t = O(\log_2 p)$ and p is small there is not a significant impact on the resulting CNF problem due to p and t.

If the SAT problem is unsatisfiable, we need to add a certain number of redundant numbers of columns in matrix D so as to satisfy the condition in (5.3) and conditions on

the decoder matrix (5.4). Suppose we wanted to test if by adding k extra columns to D we can get a feasible decoder matrix. This will only cause the dimensions of D and H to change and the Boolean conditions presented in (5.3), (5.4) remains valid and can be applied to get the new decoder matrix H as well as the redundant column entries.

If in addition D contains some don't cares, this will increase the likelihood of redundant column reduction, by proper assignment of don't cares. The assignment problem for don't cares has been incorporated in the framework of the formulation presented before. We can always get a feasible decoder matrix by adding $\triangle(p)$ redundant columns using traditional hamming code method according to the lemma stated below:

Lemma: There exists a matrix $X_{l\times\triangle(p)}$ such that $[D\ X]_{l\times(p+\triangle(p))}$ admits a feasible decode matrix $H_{\triangle(p)\times(p+\triangle(p))}$. The matrix H is constructed by keeping in its columns $\triangle(p)$ bit long Boolean representation of the numbers starting from 1 to $(p+\triangle(p))$ with the power of 2 coming in end in increasing order and the rest of the numbers coming before also in increasing order.

Thus we start in an incremental fashion by adding 1 extra redundant column and checking the feasibility of decode matrix by solving (5.3). The procedure is guaranteed to terminate by Lemma stated before. Since $\triangle(p) \sim O(\ln p)$, the run-time overhead is small for this iterative approach.

5.5 Experiments and Probabilistic Analysis for Yield Improvement

We first studied the extent to which it is possible to reduce the addition of redundant columns in code construction. For this setup, we considered LUTs of sizes 2^3x3 (3 inputs, 3 outputs) and 2^4x4 (4 inputs, 4 outputs).

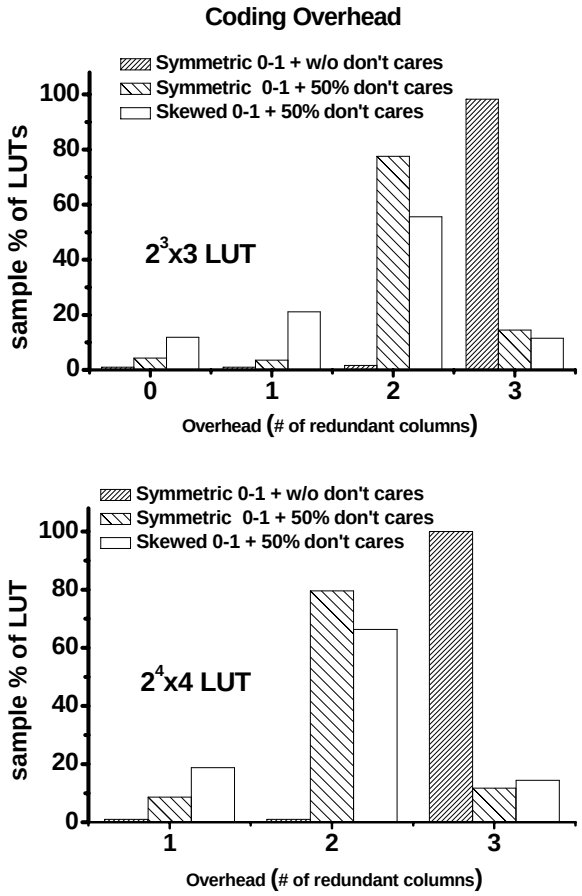

Figure 5.2 Coding overhead in terms of extra columns. Don't cares significantly increase fraction of LUTs requiring smaller overhead.

The experiments were conducted using 5000 randomly generated 0-1 matrices. We considered two scenarios: "symmetric" - in which the probability of 0 and 1 were equal, and "skewed" - in which the probability of 1 was 0.2. The experiments were repeated after introducing don't-care conditions into LUTs, with the number of don't cares set to 50% of all entries, which seems to be a reasonable number.

The max number of extra bits required by the Hamming code is 3 for both matrices. Figure 5.2 shows that in the case of 16×4 LUT, almost all LUTs required the

maximal number of redundant columns 3 in the absence of don't cares. Notably, introducing the don't cares allowed 80% of all LUTs to require only 2 redundant columns. Experiments also indicate that symmetric matrices are more difficult to encode - skewed LUTs permitted a higher reduction in the number of redundant columns. We estimated the reduction of area compared to naïve coding by defining the area of an LUT to be the number of bits in it that includes the redundant bits. The use of don't cares allows us to reduce the area by 23%, on average, and savings range from 16% to 34% across the LUT's ranging from 1-input, 1-output to the case of 4-inputs, 4-outputs.

We also studied improvements enabled by the single bit error correction. Consider an LUT with m rows and n columns. Let p be the *defect probability* of a single bit. Let the defect probability of individual bits be independent. A single row consisting of n bits is error free with probability of $(1-p)^n$. Now if there are $f(n)$ rows of size n in a circuit, then the probability of the error free circuit is $\psi(n) = (1-p)^{nf(n)}$.

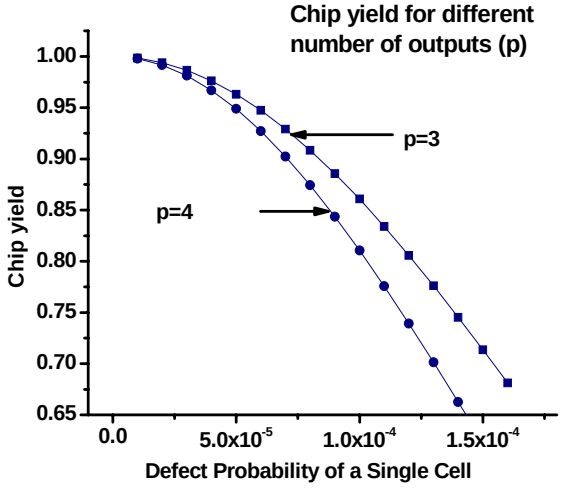

Figure 5.3 Our coding strategy allows achieving good chip yields. Yield without error correction is effectively 0. The number of LUT's is 2^{16}.

Suppose we include single bit error correction and add $s(n)$ redundant bits. In the worst case, $s(n)$ is always bounded by $\triangle(n)$. For the case of single bit error correction, the probability of an error free single row of n-bits becomes

$$\chi(n) = (1-p)^{n+s(n)} + (n+s(n)-1)(1-p)^{n+s(n)-1}p$$

For the entire circuit the error free probability is $\chi(n)^{f(n)}$. In general, we may have LUTs with different numbers of outputs; thus the overall yield probability will be the product of $\psi(n)$ or $\chi(n)$ over all feasible n in the case of no error correction, and single error correction, respectively.

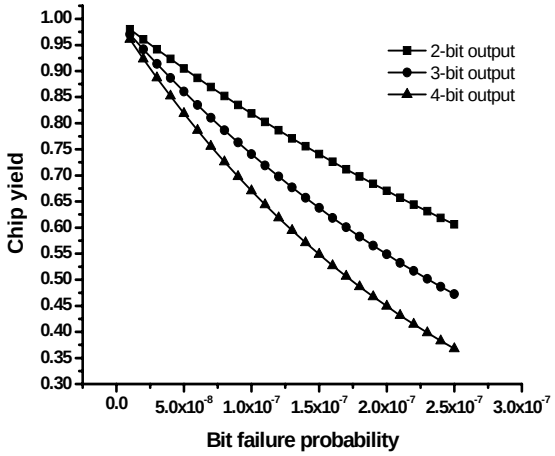

Figure 5.4 Yield without any error correction. For the range of failure probability shown, yield with a single bit error correction remained above 99%.

Assuming that we always use the default number of redundant columns, we estimate the yield for the LUT with outputs of size 3 and 4. We assume that there are 2^{16} blocks where each block is a single LUT with 16 rows. We varied the bit failure probability from 1e-5 to 16e-5. The yield with no error correction remained smaller that 1e-9 in the all the cases. With our single-bit error correction, yield remained above 70% for most of the cases, Figure 5.3. Without error correction, reasonable yield can be

expected for defect densities that would have to be 1000X smaller. In another experiment we varied single bit failure probability from 1e-8 to 25e-8. In this range of failure probabilities the yield with a single bit error correction remains always above 99%, Figure 5.4.

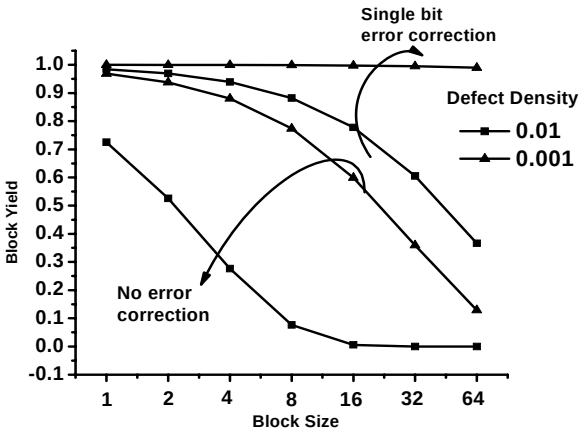

Figure 5.5 Our error correction scheme raises significantly the block yield, i.e. the probability of instantiation even for very high defect density (.01)

Finally, we studied the amount of yield improvement permitted by our error correction for different block sizes and defect densities which are projected for nano-technologies. We considered fairly high defect densities of 0.01 and 0.001. Such defect densities (probabilities) may be expected in nano-technologies. The results are for LUTs with 2 inputs and 2 outputs, Figure 5.5. We can observe that for defect density of 0.01, the yield in the case of single-bit error correction dropped from 99% to 40% with block size increase. If we do not apply error correction, yield quickly drops to nearly 0% yield when the number of blocks increases to 8. The yield improvement for block of sizes of 16 and more is higher than 30%. For the defect density of 0.001, the yield with a single bit error correction remained more than 99% across all block sizes while the yield without error correction drops to less than 10%.

5.6 CONCLUSION

We introduced a methodology for realizing coded Boolean functions implemented in nano-gates. We showed that this coded implementation along with the CMOS decoder allows us to increase the yield of nano-circuits significantly in presence of high defect density. We propose novel extensions to the Hamming code techniques to reduce the coding overhead using the special structure of Boolean functions and the presence of don't cares. We also described a heterogeneous circuit fabric and architecture in which such a coded implementation of Boolean function can be effectively realized.

Chapter 6. Path-Based Statistical Static Timing Analysis

Timing analysis is another significant design step impacted by the process variation. The parametric variation makes the circuit timing a random variable, whose distribution needs to be identified in order to predict yield. These variations exhibit themselves as *systematic*, *spatial* and *random* changes [1] in the parameters of active (transistor) and passive (interconnect) components. Furthermore, these variations are increasing with each new generation of technology. *Statistical Static Timing Analysis* (SSTA) has been proposed to perform full-chip analysis of timing under such types of uncertainty, and has been the subject of intense research recently. The result of SSTA is the prediction of parametric yield at a given target performance for a design. SSTA algorithms can be classified into two major groups:

Block-based [38][39][40][41][42][43][49][50][51][52][53][54] approaches use a breadth-first traversal of the circuit to compute circuit delay. The delay pdf is propagated from the primary inputs to the primary outputs. The major difficulty in block-based approaches is the introduction of the *max* operator at each block, and the need to accurately estimate the maximum of two random variables in the same form in which those two variables are defined.

Path-based [44][45][46][47][48] approaches rely on an enumeration of all or a large number of the most critical paths in the circuit. Considering the case where all paths are enumerated, the max operator is deferred to the end of the analysis (i.e. taking the maximum of all the paths) and therefore does not introduce any inaccuracy in the computation. A major problem with path-based approaches is the perception that typical circuits have an exponential number of paths, making the computational requirement for such approaches impractical.

While there has been much work on the algorithms for SSTA, the existing solutions still lack required accuracy. Some of the sources of inaccuracy in SSTA are: (a) the basic assumptions underlying static timing analysis, such as ignoring the functionality of gates which gives rise to false paths, (b) the delay models used for gates and wires, and (c) the models for process variations and their spatial and/or temporal distributions.

The algorithmic error introduced by SSTA algorithms can be traced back to the application of the *max* operator, which is an approximation to the behavior of true circuits, and which is further approximated in SSTA algorithms. While a direct assessment of that error is difficult, we propose that minimizing the number of max operations would aid in reducing the error. The algorithm we propose in this work reduces the number of max operations to one per circuit.

The proposed method starts with a preprocessing step of path enumeration and delay computation of the library gates in a parameterized form, which is represented using a sparse matrix. Next, the path delays in parameterized form are computed by a natural extension to the gate delay formulation. In the Monte Carlo simulation the path delays samples are obtained by multiplying the path delay coefficient of the parameterized form which are stored in a sparse matrix with entries drawn from a multivariate distribution of process parameters. Finally, a distribution of circuit delay is obtained by applying the max operator on the path delays.

The advantage of path-based SSTA is that it can naturally handle accurate nonlinear delay models. We presented a parameterized gate delay model which explicitly takes slope propagation into account. In current published approaches, typically worst-case estimate of the slope or the latest arriving slope is propagated [18] which can lead to significant error [56]. By modeling the input slope in the gate delay equation we avoid this modeling error.

The major attributes of this work are:

1) It can handle global, spatial and intra-die variations in one unified framework.
2) It can compute the delay based on an accurate propagation of slope along all the paths.
3) It is independent of the underlying distribution of the process parameters, not restricted to the usual Gaussian distribution.

6.1 STATISTICAL STA BASED ON SPARSE-MATRIX OPERATIONS

6.1.1. Parameterized Gate and Path Delay Modeling

In order to generate the cell delay model for every gate in the library, we simulate each gate varying the process parameters uniformly in the range of $\mu \pm 3\sigma$ with $3\sigma = 0.2\mu$ s. The load capacitance C_L and input slope S_{in} were also varied. The samples of S_{in} were generated in the range of 10 to 100 ps and samples of C_L were generated in the range of 1 to 10 fF. Then the values were fit (least square regression fitting) to the delay equation given below: (The functions f_1 and f_2 are quadratic while f_3 is linear)

$$D = f_1(L, V_{th}) + C_L f_2(L, V_{th}) + S_{in} f_3(C_L)$$

Similarly, the output slope was also fit to the same canonical form as delay. Note that both the output delay equation (2) and the output slope equation are explicitly dependent on input slope S_{in}.

In this and next sections, we present the sparse-matrix based SSTA formulation. First, we calculate the path delays without considering slope propagation and in the next section we take the slope into account.

6.1.2 Efficient Path-Delay Computation

The path delay is given simply by the multiplication of the path-gate incidence matrix[2] with the gate delay vector:

$$d_{path} = A d_{gate} \quad (6.1)$$

The overall circuit delay is given by the max of all path delays:

$$d_{circuit} = \max(d_{path}) \quad (6.2)$$

We note that STA in this form is essentially a sparse matrix vector multiplication, and that it requires only a single max operator to find the circuit delay. There are many data structures and algorithms developed for efficient sparse matrix manipulation which we can exploit [57]. Now we turn our attention to the Statistical STA (SSTA).

Let the delay of gate j be a function of r parameters $z_j \in R^r$ as shown below, c_j is a constant coefficient vector.

$$d_j = \sum_{k=1}^{r} c_{jk} z_{jk} = c_j^{\mathrm{T}} z_j \quad (6.3)$$

The gate delays of a circuit can be written as,

$$d_{gate} = C^{\mathrm{T}} Z \quad (6.4)$$

The path delays are obtained by multiplying the path-gate incidence matrix in (6.1) and the gate delay vector in (6.4)

$$\begin{aligned} d_{path} &= A d_{gate} \\ &= A C^{\mathrm{T}} Z \end{aligned} \quad (6.5)$$

If the process parameter vector due to *k*-th random sample is given by $Z^{(k)}$ gate then path delay vector is given by

[2] A matrix which has rows corresponding to the paths and columns corresponding to the gates. The matrix entry in row corresponding to path *p* and column corresponding to the gates *g* is 1 if *g* is on *p* otherwise 0

$$d_{path}^{(k)} = A C^{\mathrm{T}} Z^{(k)} \quad (6.6)$$

Now if we take l samples then (6.6) can be generalized as

$$[d_{path}^{(1)} \quad \cdots \quad d_{path}^{(l)}] = A C^{\mathrm{T}} [Z^{(1)} \quad \cdots \quad Z^{(l)}] \quad (6.7)$$

To get the circuit delay distribution, we apply (6.3) to Eq. (6.7)

$$[d_{circuit}^{(1)} \quad \cdots \quad d_{circuit}^{(l)}] = [\max(d_{path}^{(1)}) \quad \cdots \quad \max(d_{path}^{(l)})]$$

This is essentially a Monte Carlo simulation expressed in matrix form. A histogram of the circuit delay vector produces the circuit delay distribution.

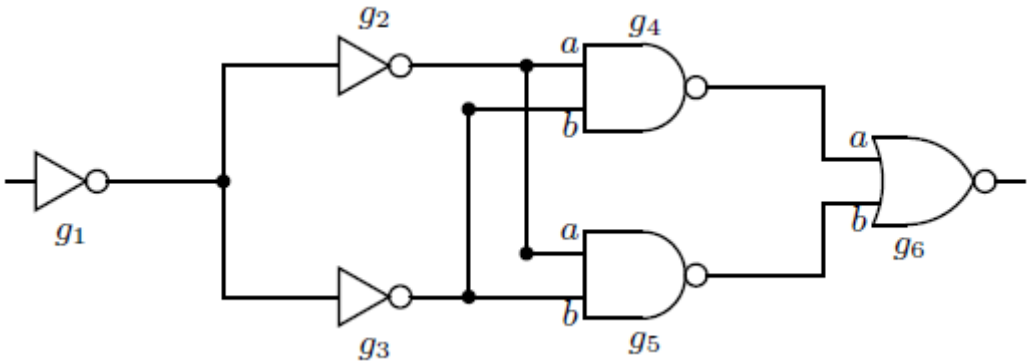

Figure 6.1 Example circuit for illustrating the matrix formulation. For 2-input gates, the input pins are identified by the labels a and b.

6.1.3 Handling of Slope Dependency

We now extend our delay model to include slope propagation. It is important to note that the output slope of gate j cannot be specified unless we know which path it belongs to. For example, in Figure 6.1, gate g_4 will have two different slopes namely:

1. s_{14} due to path 1 ($g_1 \rightarrow g_2 \rightarrow g_{4a} \rightarrow g_{6a}$), and
2. s_{44} due to path 4 ($g_1 \rightarrow g_3 \rightarrow g_{4b} \rightarrow g_{6a}$).

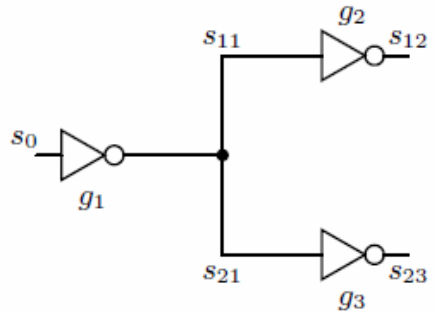

Figure 6.2 A simple circuit to illustrate SSTA with slope propagation. Heres0 denotes the slope at the primary input. The output slope at gate g1 in path 1 is denoted as s11 and in path 2 is denoted as s21.

We use the same canonical form to express both delay and slope, but we restrict the dependence of delay and output slope on the input slope to be linear. This linearity is required in order to preserve the canonical form as delays are accumulated along a path. We use the superscripts d and s to distinguish among them. We delineate the input slope to a gate by the subscript in. The gate delay d_{ij} and the output slope s_{ij} of gate j in path i is given by:

$$d_{ij} = \lambda_j^d s_{in} + w_j^d \quad (6.8)$$

$$s_{ij} = \lambda_j^s s_{in} + w_j^d \quad (6.9)$$

Where the term ω is independent of slope and can be expressed in the canonical form of (6.3), i.e. $\omega = c^{\mathrm{T}} z$. The input slope at all the gates is required to calculate the gate and path delays. One way to solve for the input slope is to look at each path p separately and obtain the slope of each gate in an individual path. This method is illustrated using the Figure 6.2, and this simple circuit consists of inverters which allow us to conveniently drop the input-pin specific subscripts.

To illustrate, consider the path p = 1, through gates g1 and g2 in Figure 6.2. The column vector s1 is given by

$$s_1 = \begin{bmatrix} s_0 \\ s_{11} \\ s_{12} \end{bmatrix}$$

and related by (6.9) as

$$\begin{bmatrix} s_0 \\ s_{11} \\ s_{12} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ \lambda_1^s & 0 & 0 \\ 0 & \lambda_2^s & 0 \end{bmatrix} \begin{bmatrix} s_0 \\ s_{11} \\ s_{12} \end{bmatrix} + \begin{bmatrix} \omega_0^s \\ \omega_1^s \\ \omega_2^s \end{bmatrix} \quad (6.10)$$

where $s_0 = \omega_0^s$.

Let s_p be the column vector in which the values of slope along path p are listed. Assume the values are listed in the topological order of the gates along path p. In general (6.10) is valid for any arbitrary path containing t gates. Thus, $s_1 \in R^{t+1}, \Lambda_1^s \in R^{(t+1)\times(t+1)}$ is lower diagonal, and $\omega_1^s \in \mathbb{R}^{(t+1)}$. If the circuit has p paths, then the (6.10) for all the p paths can be succinctly captured into one single equation shown below

$$\begin{bmatrix} s_1 \\ \cdots \\ s_p \end{bmatrix} = diag(\Lambda_1^s, \ldots, \Lambda_p^s) \begin{bmatrix} s_1 \\ \cdots \\ s_p \end{bmatrix} + \begin{bmatrix} \omega_1 \\ \cdots \\ \omega_p \end{bmatrix}$$
$$s = \Lambda^s s + \omega^s \quad (6.11)$$

From (6.11) we can solve for the slope s in the circuit

$$s = (I - \Lambda^s)^{-1} \omega^s$$

Lemma 1: The matrix $(I - \Lambda^s)$ is non-singular. Thus its inverse exists.

The equation for gate delays is similar to (6.11) and can be generalized to make the gate delay a function of parameters.

$$d_{gate} = \Lambda^d s + \omega^d = \Lambda^d (I - \Lambda^s)^{-1} \omega^s + \omega^d$$
$$= (\Lambda^d (I - \Lambda^s)^{-1} (C^s)^{\mathrm{T}} + (C^d)^{\mathrm{T}}) Z = DZ$$

Once the gate delays are calculated, we can find the path delays using (6.7). The circuit delay is given by the max of all path delays. Since delay and slope are a function of process parameters, by taking samples of process parameters one can generate samples of circuit delay. A histogram on these samples gives us the circuit delay distribution. Thus SSTA can be performed considering the slope and process variations.

6.2 EXPERIMENTS

We implemented our algorithm using a combination of awk/perl scripts and C++. We report the results of experiments run on the ISCAS'89 benchmarks using a 64-bit Linux machine with 16 GB RAM and running at 3.4 GHz. The delay models were generated using the 90 nm Berkeley Predictive Technology Model. We modeled the effect of variations in channel length and threshold voltage, and assumed that the variance of these parameters was such that $3\sigma = 0.2\mu$.

We performed 10000 Monte Carlo simulations for each of the ISCAS benchmark circuits. The results are summarized in Table 6.1. Note that the path generation step takes a modest portion of the overall runtime, less than 21% for smaller benchmarks and nearly 5% for bigger benchmarks, while the parameterized path delay generation, which builds the various matrices and the matrix multiplication, take the bulk of the runtime.

The relationship between number of paths and runtime is approximately

$$runtime \approx 0.006 \times paths^{1.09}$$

We compared the runtime for Monte-Carlo simulations reported in [55] noting that the machine specifications are comparable, and that the number of Monte-Carlo samples is identical. The runtimes were quite close, especially for the larger benchmarks,

in spite of (a) the use of a simpler linear delay model, (b) not accounting for slope propagation, and (c) a complete compiled code (C++) implementation in [55].

Table 6.1 Path-gate statistics of ISCAS89 benchmarks and runtime for Monte Carlo.

Circuit	paths	Gates	Sparsity [%]	Runtime [s]				Percentage runtime (%)			Tme per matrix multiply [s]
				Generating		Matrix multiply	Total	Generating		Matrix multiply	
				Paths	matrix			paths	matrix		
s713	250	2650	17.54	5.13	36.47	50.42	92.02	5	39	54	5.00e-03
s5378	1938	6858	0.66	4.44	9.62	20.28	34.34	12	28	59	2.00e-03
s1432	566	35990	5.61	37.42	255.02	384.52	676.96	5	37	56	3.80e-02
s35932	14773	122997	0.16	109.93	436.57	1041.31	1587.81	6	27	65	1.03e-01
s9234	5158	227837	0.74	319.25	2030.37	4236.31	6585.93	4	30	64	4.20e-01
s38584	15351	850422	0.25	1121.40	6925.62	11612.28	19659.30	5	35	59	1.15
s13207	7070	1005680	0.63	1575.20	10385.31	15364.06	27324.57	5	38	56	1.52
S38417	21633	1389348	0.13	1349.61	5779.38	13128.55	20257.54	6	28	64	1.30

6.3 CONCLUSION

This work demonstrates that it is possible and practical to perform path based statistical static timing analysis, and that such an analysis can be written compactly in matrix notation, allowing the use of standard highly optimized linear algebra techniques. The major advantage of this formulation is that it embeds accurate polynomial-based delay model which takes into account slope propagation naturally. With the exception of the need to have the slope appear linearly, fairly arbitrary models can be trivially handled

using this framework. Data was presented to show that many practical circuits have a bounded number of paths, making such an analysis possible. It should be noted that this demonstration should not be taken as sufficient license to propose a purely path-based SSTA algorithm. For example, the s15850 ISCAS89 benchmark circuit had , 150×10^6 paths and could not be handled. We plan to explore efficient non-critical path removal techniques to reduce the matrix sizes.

Chapter 7. Summary

In this Ph.D. dissertation, we developed a set of analysis and optimization algorithms for statistical circuit synthesis and analysis. First we presented the synthesis algorithm that performs maximization of timing and power-limited parametric yield taking into account the process variations. The algorithm implements a statistical version of technology mapping for power under delay constraints. Across the benchmarks the algorithm achieves significant reductions in leakage power. This is the first practical demonstration of the possibility of rigorous statistical synthesis in the presence of variability. The future work will aim at enhancing the capacity and improving the run-time of the algorithm.

Second, we have described the first gain based technology mapper which can accurately capture the leakage cost estimates during simultaneous propagation of delay and leakage-slope trade-off points. A key contribution is that the leakage of any partial mappings can be shown to have a linear dependency on the output capacitance. Thus in spite of unavailability of capacitance, we can make accurate choices of leakage-inferior mappings at the fanin cones of intermediate subject graph vertices based on the slope of leakage with respect to the output capacitance. We also extended the algorithm to utilize the state dependency of the leakage for better leakage optimization. We demonstrated the efficiency of our algorithm in contrast to the previous gain based mappers within our own framework, we also contrasted the dominant-leakage based optimization with the state probability aware penalty function based mapper and showed the need for probability aware optimization. Future work includes extension of the algorithm to target the

discrete-size cell library and developing some heuristics for ideal choices of the global gain.

Third, we have also considered extension of the above gain-based approach to the scenario when the input state probabilities are not point probabilities, but rather are uncertain, and best modeled by an interval. In this case, we define the notion of a proxy metric, called robust leakage, which is sum of the worst mean value of leakage of all the circuit gates under all possible scenarios of the input state probabilities. We proposed a technology mapping algorithm to minimize the robust leakage. The algorithm is based on the application of robust dynamic programming. We compared the mappings produced by the robust mapper to the mappings of the point probability mapper and showed that we can improve leakage if the point probability estimates are not accurate. Thus, the algorithm enables us to reduce the sensitivity of the optimization result to a specifically chosen point probability.

Fourth, we addressed the redundancy, or area overhead, for realizing coded Boolean functions implemented in nano-gates which are characterized by high defect density. We proposed novel extensions to the Hamming code based on the special structure of Boolean functions and the presence of don't cares. We also described a heterogeneous circuit fabric and architecture in which such a coded implementation of Boolean function can be effectively realized.

Finally, we demonstrated that it is possible and practical to perform path-based statistical static timing analysis, and that such an analysis can be written compactly in matrix notation, allowing the use of standard highly optimized linear algebra techniques. The major advantage of this formulation is that it places no restrictions on process parameter distributions. It embeds an accurate polynomial-based delay model which naturally takes into account slope propagation. With the exception of the need to have the

slope appear linearly, fairly arbitrary models can be trivially handled using this framework. The future work is to explore efficient non-critical path removal techniques to reduce the matrix size. We may employ a heuristic for identifying near critical paths in circuits containing more than a million paths and pass only the near-critical paths to the algorithm.

References:

[1] D. Boning and S. Nassif, "Models of process variations in device and interconnects," *Design of high-performance microprocessor circuits,* A. Chandrakasan et al., Chapter 6.

[2] V. Zolotov et al., "Computation of yield gradients from statistical timing analysis," *In International workshop on timing issues in the specification and synthesis of digital systems*, 2006.

[3] S. Borkar *et al.,* "Parameter variation and impact on Circuits and Microarchitecture," *Proc. of DAC*, 2003, pp. 338-342

[4] Q. Wang, and S. Vrudhula, "Static power optimization of deep submicron CMOS circuit for dual V_{th} technology," *Proc. of ICCAD*, 1998, pp. 490-496.

[5] D. Lee *et al.*, "Simultaneous State, Vt and Tox Assignment for Total Standby Power Minimization," *Proc. of DATE*, 2004, pp. 494-499.

[6] P. Gupta *et al*., "Selective Gate-Length Biasing for Cost-Effective Runtime Leakage Control," *Proc. of DAC*, 2004, pp. 327-330.

[7] C. Visweswariah, "Death, Taxes and Failing Chips," *Proc. of DAC*, 2003, pp. 343-347.

[8] B. Hu *et al.*, "Gain-based technology mapping for discrete-size cell libraries," *Proc. of DAC*, 2003.

[9] D. Jongeneel *et al*., "Area and search space control for technology mapping," *Proc. of ASP-DAC*, 2000, pp. 86-91.

[10] S. H. Gunther *et al*, "Managing the Impact of Increasing Microprocessor Power Consumption," *Intel Technology Journal*, Q1, 2001.

[11] C. Kang and M. Pedram, "Tehcnology mapping for low leakage power and high speed with hot carrier effect consideration," *Proc. of ASPDAC*, 2003.

[12] M. Iyer *et al.*, "Wavefront technology mapping," *International Workshop on Logic synthesis*, 1998.

[13] D. Lee and D. Blaauw, "Static leakage reduction through simulataneous threshold voltage and state assignment," *Proc. of DAC*, 2003.

[14] P. Kudva *et al*., "Gain-based logic synthesis," *Proc. of ICCAD tutorial*, 2000.

[15] R. Rudell, "Logic synthesis for VLSI design," *PhD. Thesis, University of California, Berkeley*, 1989.

[16] A. Davoodi *et al.*, "Variability inspired implementation selection problem," *Proc. of ICCAD*, 2004, pp. 423-427.

[17] S. Roy *et al.*, "An alpha-approximate algorithm for delay-constraint technology mapping," *Proc of DAC*, 1999, pp. 367-372.

[18] C. Visweswariah et al., "First-order incremental block-based statistical timing analysis," Porc. of DAC, 2004, pp. 331-336.

[19] A. Srivastava, *et al*., "Statistical optimization of leakage power considering process variations using dual-V_{th} and sizing," *Proc. of DAC*, June 7-11, 2004, pp. 773 – 778.

[20] P. Seung *et al.*, "Novel sizing algorithm for yield improvement under process variation in nanometer technology", *Proc. of DAC*, 2004, June 7-11, 2004, pp. 454 – 459.

[21] M. Mani and M. Orshansky, "A new statistical optimization algorithm for gate sizing," *Proc. of ICCD*, 2004, pp. 272 – 277.

[22] A. Srivastava et al., "Accurate and efficient gate-level parametric yield estimation considering correlation variations in leakage power and performance," *Proc. of DAC*, 2005, pp. 535-540.

[23] H. Chang and S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single pert-like traversal," Proc. of ICCAD, 2003, pp. 621-625.

[24] S. Borkar et al., "Parameter variation and impact on circuits and microarchitecture," *Proc. of DAC*, 2003.

[25] R. Rudell, "Logic synthesis for VLSI design," *PhD. Thesis*, University of California, Berkeley, 1989.

[26] J. Grodstein *et al.*, "A delay model for logic synthesis of continuously-sized networks,"*Proc. of ICCAD*, 1995.

[27] D. Blaauw et al., "Slope propagation in static timing analysis," *IEEE transactions on Computer-Aided Design of Integrated Circuits*, 2002, vol. 21, no. 12.

[28] R. Rao *et al.*, "Parametric yield estimation considering leakage variability," *Proc. of DAC*, 2004, pp. 442-447.

[29] A. Emrah *et al*., "Leakage and leakage sensitivity computation for combinational logic," *Journal of Low Power Electronics*, August 2005, pp.172-181.

[30] A. Aziz *et al.*, "Sequential synthesis using SIS," *IEEE Trans. on CAD*, 2000, pp. 1149-1162.

[31] M. Orshansky and A. Bandyopadhyay, "Fast statistical timing analysis handling arbitrary delay correlations," *Proc. of DAC*, 2004, pp 337-342.

[32] S.-J. Kim et al., "A Heuristic for Optimizing Stochastic Activity Network with application to Statistical Circuit Sizing", *Operational Research*, 2005, pp. 899-932.

[33] E. Horowitz and S. Sahni, "Fundamentals of computer algorithms," *Computer Science Press*, 1978.

[34] I. Sutherland and R. Sproull, "The theory of logical effort: designing for speed on the back of an envelope," *Advanced Research in VLSI*, 1991.

[35] K. Keutzer, "Technology binding and local optimization by DAG mapping," *Proc. of DAC*, 1987.

[36] K. Chaudhary and M. Pedram, "A nearly optimal algorithm for Technology Mapping minimizing area under delay constraint," *Proc. of DAC,* 1992.

[37] Y. Kukimoto *et al*, "Delay-optimal technology mapping by DAG covering," *Proc of DAC.*

[38] Jing-JiaLiou et al., "Fast statistical timing analysis by probabilistic event propagation," *Proc of DAC*, 2001, pp. 661-666.

[39] Aseem Agarwal et al., "Computation and refinement of statistical bounds on circuit delay," *Proc of DAC*, 2003, pp. 348-353.

[40] Anirudh Devgan and Chandramouli Kashyap, "Block-based static timing analysis with uncertainty," *Proc of ICCAD*, 2003, pp. 607-614.

[41] Hongliang Chang and Sachin S. Sapatnekar, "Statistical timing analysis under spatial correlations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2005, pp. 1467-1482.

[42] C. Visweswariah et al., "First-order incremental block-based statistical timing analysis," *Proc of DAC*, 2004, pp. 331-336.

[43] Jiayong Le et al., “STAC: statistical timing analysis with correlation,” *Proc of DAC*, 2004, pp. 343-348.

[44] Anne E. Gattiker et al., “Timing yield estimation from static timing analysis,” *Proc of ISQED*, 2001, pp. 437-442.

[45] Jing-Jia Liou et al., “False-path-aware statistical timing analysis and efficient path selection for delay testing and timing validation,” *Proc of DAC*, 2002, pp. 566-569.

[46] Aseem Agarwal et al., “Path-based statistical timing analysis considering inter and intra-die correlations,” *Proc of International Workshop on Timing Issues*, 2002.

[47] J. A. G. Jess et al., “Statistical timing for parametric yield prediction of digital integrated circuits,” *Proc of DAC*, 2003, pp. 932-937.

[48] Michael Orshansky and Arnab Bandyopadhyay. Fast statistical timing analysis handling arbitrary delay correlations. *Proc of DAC*, 2004, pp. 337-342.

[49] Yaping Zhan et al., “Correlation-aware statistical timing analysis with non-gaussian delay distributions,” *Proc of DAC*, 2005, pp. 77-82.

[50] Lizheng Zhang et al., “Correlation-preserved non-gaussian statistical timing analysis with quadratic timing model,” *Proc of DAC*, 2005, pp. 83-88.

[51] Vishal Khandelwal and Ankur Srivastava, “A general framework for accurate statistical timing analysis considering correlations,” *Proc of DAC*, 2005.

[52] Hongliang Chang et al., “Parameterized block-based statistical timing analysis with non-gaussian parameters, nonlinear delay functions,” *Proc of DAC*, 2005, pp. 71-76.

[53] Khaled R. Heloue and Farid N. Najm, “Statistical timing analysis with two-sided constraints,” *Proc of ICCAD*, 2005, pp. 829-836.

[54] Debjit Sinha and Hai Zhou, “A unified framework for statistical timing analysis with coupling and multiple input switching,” *Proc of ICCAD*, 2005, pp. 837-843.

[55] Jaskirat Singh and Sachin S. Sapatnekar, “Statistical timing analysis with correlated non-gaussian parameters using independent component analysis,” *Proc. of International Workshop on Timing Issues*, 2006.

[56] David Blaauw et al., “Slope propagation in static timing analysis,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2002, pp. 1180-1192.

[57] Yousef Saad, “Iterative Methods for Sparse Linear Systems,” *Society for Industrial and Applied Mathematics*, 2003.

[58] Ashish Kumar Singh et al., “Statistical technology mapping for parametric yield,” Proc of ICCAD, 2005.

[59] Ashish Kumar Singh et al., “Gain-based technology mapping for minimum runtime leakage under input vector uncertainty,” Proc. of DAC, 2006.

[60] Anand Ramalingam et al, “An accurate matrix based formulation for statistical static timing analysis,” Proc of ICCAD, 2006.

[61] T. I. Kamins and R. S. Williams, “Trends in nanotechnology: Self-assembly and defect tolerance,” *Proc. NSF Partnership in Nanotechnology Conf.*, 2001.

[62] R. H. Baughman, A. Zakhidov, W. A. de Heer, “Carbon Nanotubes-the route toward applications,” *Science*, pp. 787-792, 2002.

[63] J. Chen, "Carbon Nanotube Electronics and Optoelectronics", *Embedded Tutorial: Emerging Nanoelectronics: Prospects, State of the Art, and Opportunities for CAD, Proc. of ICCAD*, 2006.

[64] B.W. Johnson, “Fault Tolerance,” *The Electrical Engineering Handbook*, R. C. Dorf, ed., CRC Press, pp. 2020, 1993.

[65] M. Banatre and P. Lee, Hardware and Software Architecture for Fault Tolerance: Experience and Perspective, Springer-Verlag LNCS, Vol. 774, 1994.

[66] J. von Neumann, “Probabilistic logics and the synthesis of reliable organisms from unreliable components," *Automata Studies*, C. E. Shannon and J. McCarthy Eds., Princeton University Press, Princeton, NJ, 1956, pp. 329-378.

[67] N. Pippenger, “On networks of noisy gates," *Proc. of 26-th IEEE Symposium on the Foundations of Computer Science*, 1985, pp. 30-36.

[68] K. Nikolic, A. Sadek, and M. Forshaw, "Fault-tolerance techniques for nanocomputers," *Nanotechnology*, 2002, pp. 357-362

[69] D. Strukov, K. Likharev, "CMOL FPGA: a reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices", *Nanotechnology*, 1005, Volume 16, Issue 6, pp. 888-900.

[70] S. Lin and D. J. Costello, Error Control Coding, Prentice Hall, 2004, 2nd edition.

[71] P. K. Kuekes, W. Robinett, G. Seroussi and R. S. Williams, "Defect-tolerant interconnect to nanoelectronic circuits: internally redundant demultiplexers based on error-correcting code," *Nanotechnology*, 2005, pp. 869-892.

[72] B. S. Tsybakov, S. I. Gel'fand, A. V. Kuznetsov, and S. I. Ortyukov, "Reliable computation and reliable storage of information," *IEEE-USSR Workshop Proc.*, 1975.

[73] S.I. Gelfand and M.S. Pinsker, "Coding for channel with random parameters" *Problems of Control and Information Theory*, 1980, pp-19-31.

[74] W.W. Paterson and M. O. Rabin, "On codes for checking logical operations," *IBM J. of Research and Development*, 1959, No. 4.

[75] G. Tstein, "On the complexity of derivation in propositional calculus," *Studies in constructive Mathematics and Mathematical Logic, part 2(1968),* pp. 115-125.

[76] Giovanni De Micheli, Synthesis and Optimization of Digital Circuits, Mc-Graw-Hill, 1994.

[77] N. Eén, N. Sörensson, "An extensible SAT-solver", SAT 2003.

[78] K. K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation, Wiley, 1999.

[79] Y. Watanabe, L. Guerra, R. Brayton, "Permissible functions for multioutput components in combinational logic optimization," *IEEE Trans. on CAD*, 5(7): 732-744, 1996

[80] S. Sirichotiyakul *et al.*, "Duet: an accurate leakage estimation and optimization tool for dual-Vt circuits," *IEEE Trans. on VLSI Systems*, 2002, vol. 10, pp. 79-90.

[81] Y. Taur et al., "CMOS scaling into the nanometer regime," *Proc. of the IEEE*, no. 4, 1997, pp. 486-504.

[82] R. Brodersen et al., "Methods for True Power Minimization," *Proc. of ICCAD*, 2002, pp. 35-40.

[83] D. Lee and D. Blaauw, "Static leakage reduction through simultaneous threshold voltage and state assignment," *Proc. of DAC*, 2003.

[84] D. Lee et al., "Runtime Leakage Minimization through Probability-Aware Dual-Vt or Dual-Tox Assignment," *Proc. of ASP-DAC*, 399-404, January, 2005.

[85] F. Najm, "Transition density, a stochastic measure of activity in digital circuits," *Proc. DAC,* pp.644-649, 1991.

[86] S. M. Kang, “Accurate Simulation of Power Dissipation in VLSI Circuits”, *IEEE Journal of Solid-state Circuits*, vol. SC-21, no. 5, pp.889-891, Oct. 1986.

[87] F. N. Najm et al., “Probabilistic Simulation for Reliability Analysis of CMOS Circuits”, *IEEE Transaction on CAD*, vol 9-4, pp.439-450, April 1990.

[88] V. Tivari et al., “Technology mapping for low power in logic synthesis”, *Journal of VLSI Integration*, vol. 20, no. 3, pp. 145-148, 1996.

[89] C. Kang and M. Pedram, “Technology Mapping for Low Leakage and High Speed with Hot Carrier Effect Consideration,” *Proc. of IWLS*, 2002, pp. 295-300.

[90] A. Nilim and L. Ghaoui, “Robust Control of Markov Decision Processes with Uncertain Transition Matrices,” *Journal of Operationsl Research*, vol 53, no. 5, pp. 780-798, 2005.

[91] L. El Ghaoui et al., “Robust solutions to uncertain semidefinite programs,” *SIAM Journal of Optimization*, vol. 9, no. 1, pp. 33-52, 1998.

[92] M. Dyer and L. Stougie, “Computational complexity of stochastic programming problems,” http://www.win.tue.nl/bs/spor/2003-20.pdf, 2004.

[93] D. Miller, “An Improved Method for Computing a Generalized Spectral Coefficient”, *IEEE Trans. on CAD* Mar 1998.

[94] M. A. Cirit, “Estimating Dynamic Power Consumption of CMOS Circuits”, *Proceedings of ICCAD*, 1987.

[95] D. Burger and T. Austin. "The SimpleScalar Tool Set, Version 2.0", *Technical Report 1342*, University of Wisconsin Computer Sciences Department.

[96] http://www.spec.org/

[97] A. J. KleinOsowski and D. L. Lilja, "MinneSPEC: A New SPEC Benchmark Workload for Simulation-Based Computer Architecture Research," *Computer Architecture Letters*, June 2002.

[98] L. Henrique et al., “Affine arithmetic: concepts and applications,” *Numerical Algorithms*, vol. 37, pp. 147-158, 2004.

[99] K. Parker, and E. J. McCluskey, “Probabilistic Treatment of General combinational Networks,” *IEEE Tranactions. on Computers*, vol. C-24, June, 1975.

[100] R. Marculescu et al., “Efficient Power Estimation for Highly Correlated Input Streams,” *Proc. of DAC*, pp. 628-634, 1995.

[101] M. R. Guthaus, N. Venkateswaran, C. Visweswariah and V. Zolotov, “Gate Sizing using Incremental Parameterized Statistical Timing Analysis”, *Proc. of ICCAD*, 2005.

[102] S. Sirichotiyakul et al., “Stand-by Power Minimization through Simultaneous Threshold Voltage selection and Circuit Sizing,” *Proc. of DAC*, 1993, pp. 436-441.

[103] M. Orshansky, L. Milor, L. Nguyen, G. Hill, Y. Peng and C. Hu, “Characterization of Spatial Intra-Fielf CD Variability,” in Proc. of SPIE Conference on Optical Microlithography, vol 4000, 2000.

[104] S. Mukhopadhyay, K. Roy, “Modeling and Estimation of Total Leakage Current in Nano-scaled-CMOS Devices considering the effect of Paremeter Variation,” in *Proc. of ISPLED*, 2002, pp. 64-67.

[105] A. K. Singh et al., “Generation of Efficient Codes for Realizing Boolean Functions in Nanotechnologies,” http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0703102.

Vita

Ashish Kumar Singh was born in Kanpur, Uttar Pradesh, India on 26th March, 1981, the son of Shri Shiv Kumar Singh and Shrimati Girish Kumari Singh. He received the Bachelor of Technology (B. Tech) degree in Computer Science and Engineering from the Indian Institute of Technology, Kanpur, in May, 2001. He completed his Master's from Royal Institue of Technology, Sweden in Aug 2003. He joined the department of Electrical and Computer Engineering at the University of Texas at Austin in September 2004, where he is currently a full time student.

Permanent Address: 104A/381A, Ram Bagh
Kanpur, Uttar Pradesh,
India

This dissertation was typed by the author.